FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

MobileVJ: A mobile app for a novel wearable human sensing system

Pavel Alexeenko

Mestrado Integrado em Engenharia Informática e Computação

Orientador: João Paulo Trigueiros da Silva Cunha

Coorientador: Rui Filipe Lima Maranhão de Abreu

19 de Julho de 2016

MobileVJ: A mobile app for a novel wearable human sensing system

Pavel Alexeenko

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Pedro Souto
Arguente: Jorge Cardoso
Vogal: João Paulo Cunha

---

19 de Julho de 2016

Resumo

Ao longo dos anos, as tecnologias não pararam de evoluir, e a sua influência na vida humana não ficou despercebida. As novas tecnologias fazem já parte do dia-a-dia dos indivíduos, por isso, e de uma forma geral, pode-se verificar que torna a vida de qualquer um mais facilitada e proporciona muitas melhorias no seu quotidiano.

Hoje em dia, na área de saúde existem inúmeros sensores vitais tanto como sensores ambientais ou de movimento. A maioria das vezes todos os sensores guardam os seus dados monitorizados num disco físico ou através de um dispositivo móvel. Como se pode imaginar, estes dispositivos geram milhões de dados, e distribuí-los, torna-se difícil. Visto este problema, a solução seria criar um sistema de monitorização em tempo real com a base de arquitetura dos sistemas M2M, capaz de reencaminhar os dados para os profissionais de saúde de modo a que sejam analisados os dados desejados.

O desenvolvimento do sistema de monitorização incluirá uma aplicação móvel que será desenvolvida na plataforma *Android*, onde será utilizada a tecnologia *bluetooth* para cumprir o objetivo de selecionar os dispositivos de sensores vestíveis e ler os dados recolhidos por estes, como também ao próprio *smartphone* em utilização. Esta aplicação deverá ser capaz de enviar os dados instantaneamente para um servidor, onde estes já serão reencaminhados para os profissionais de saúde e armazenados numa base de dados em que futuramente poderão ser analisados. Para testar o funcionamento do módulo será utilizada tecnologia de monitorização de saúde como o *VitalJacket* com sensores ambientais e de movimento.

Este projeto teria um grande impacto no controlo de saúde de atletas tanto a nível profissional como amador, na monitorização à distância de utentes nos hospitais ou mesmo nas profissões de risco, como por exemplo bombeiros ou polícias, de modo a que se consiga reagir no momento imediato se um dos funcionários se encontrar num estado de perigo.

Abstract

Through the years, technologies kept evolving, their influence in our lives did not pass unnoticed. New technologies are already part of our every day lives, so it's safe to say it provides a lot of improvements on our daily schedules.

Nowadays, health technology uses not only vital sensors but movements and environmental sensors as well. Most of the time all of them keep their data monitored in an Hard Drive or through a mobile device. As you can imagine, this devices generate millions of data, and distributing them, becomes hard. Given this, the solution would be creating a monitoring system in real time with a base architecture of the M2M systems, capable of sending that data to health professionals, allowing them to be analyzed as desired.

The development of the monitoring system will include an Android Mobile Application, which will use Bluetooth technology to fulfill the goal of selecting the desired sensors and read their data, as well as the smartphone in use itself. This application should be able to send the data instantaneously to a Server, where they will be distributed to the health professionals and stored on a Database that can be later analyzed. To test the module, technologies that use sensors, such as VitalJacket will be used.

This project would have a huge impact on health control of professional and amateur athletes, controlling hospital patients remotely or even in dangerous jobs such as firemen and policemen, so that the reaction is immediate if anyone is in need of help.

Agradecimentos

Antes de mais, gostaria de agradecer ao meu orientador, o Professor João Paulo Cunha, por me ter facultado a oportunidade de realizar esta dissertação e o Professor Rui Maranhão por toda a orientação disponibilizada e fornecimento de ideias sem as quais esta dissertação não teria sido possível de ser realizada.

Quero também agradecer ao investigador Gonçalo de Oliveira Pimentel por se mostrar sempre disponível para o esclarecimento de dúvidas que surgiram ao longo desta dissertação.

Agradeço também aos meus pais, que durante todo o meu percurso académico e neste importante momento em particular, me apoiaram e estiveram sempre presentes. Obrigado pela educação que me deram, por me facultarem os estudos e por me ajudarem a perceber que com vontade consigo enfrentar todas as dificuldades e perceber que nada é impossível.

Após ter vindo viver para Portugal, apareceram na minha vida amigos como o Pedro Alexandre e o Diogo Rodrigues. Quero-lhes agradecer por me terem acompanhado não só nas atividades desportivas, mas também por me terem apoiado nas várias situações que foram aparecendo na minha vida.

Agradeço ao Luís Abreu e ao Pedro Miranda que conheci no meu primeiro ano de faculdade. Nunca mais nos largámos, partilhámos grupos de trabalho cujos resultados não foram só bons projetos e sucesso nos estudos, mas também uma amizade gerada para o resto das nossas vidas.

Por fim, quero agradecer à minha namorada Maria, por ter partilhado comigo os últimos quatro anos das nossas vidas, por ter oferecido algumas das ideias espetaculares para vários trabalhos realizados durante o meu percurso académico e por ter me ajudado com a revisão da dissertação.

Pavel Alexeenko

"O tempo dura bastante para aqueles que sabem aproveitá-lo."

Leonardo Da Vinci

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	Application Programming Interface
App	Aplicação móvel
BLE	Bluetooth Low Energy
ECG	Eletrocardiograma
GPS	Global Positioning System
iOS	Sistema Operacional Móvel da Apple Inc.
REST	Representational State Transfer
SDK	Software Development Kit
VJ	VitalJacket®
IP	Internet Protocol
M2M	Machine to Machine

Capítulo 1

Introdução

1.1 Enquadramento

Esta dissertação enquadra-se no domínio das ferramentas de agregação de informação e desenvolvimento de *software*. Dado um número elevado de *hardware* existente que permite uma monitorização remota, torna-se vital a necessidade de criação de um *software* capaz de retirar informação destes dispositivos agrupando-a e adicionando outro tipo de informações para que a mesma possa ser analisada em tempo real ou mais tarde.

O uso de ferramenta que permite agregar diferentes sinais torna-se importante nas áreas de saúde, não só para os amadores que praticam desporto e conseguem controlar o seu ECG, distância percorrida e outra informação adicional, mas também permite aos profissionais de saúde controlar um ou vários utentes de uma forma remota.

O INESC TEC é o instituto de I&D, sem fins lucrativos e declarado de utilidade pública, que faz parte dos Laboratórios Associados da FCT (agência de financiamento de I&D do governo português) cuja atividade visa a Investigação Científica e o Desenvolvimento Tecnológico, como também a Consultoria e Formação Avançada, a Transferência de Tecnologia e o Lançamento de Novas Empresas de Base Tecnológica. Foi este instituto que propôs o tema desta dissertação.

1.2 Motivação

Ao longo dos últimos anos, surgiu uma variedade de sensores vestíveis de sinais vitais corretamente testados, certificados e a preços acessíveis com o intuito de disponilizar e fornecer aos seus utilizadores uma monitorização do seu estado de saúde de forma confortável e segura.

Atualmente, os smartphones possuem vários tipos de sensores que possibilitam o fornecimento de informação especificada sobre factores como a atividade física, localização e condições ambientais em que se encontra o utilizador. No procedimento de monitorização do sujeito, os fatores referidos anteriormente desempenham uma grande influência como por exemplo no caso da recolha de eletrocardiogramas há já um grande número de dispositivos capaz de fazer estudo dos dados recolhidos através das aplicações móveis para smartphones que estabelecem a interface

gráfica com os utilizadores. No entanto, os sistemas atuais compostos pelos sensores externos e pelas aplicações móveis ficam restringidos normalmente à análise da atividade cardíaca isolada esquecendo-se que esta pode ser influenciada pelos fatores externos.

Na grande parte dos sistemas de monitorização que existem, os dados são analisados diretamente nos dispositivos móveis e são demonstrados para o utilizador do mesmo dispositivo ou são enviados para os profissionais de saúde. O MobileVJ visa ser um sistema de monitorização em tempo real com qualquer tipo de dispositivo de forma a analisar os dados na parte do servidor para que haja uma diminuição do consumo dos recursos do dispositivo móvel.

O projeto MobileVJ possui particular interesse para instituto de investigação INESC TEC, uma vez que este será um sistema complexo que permitirá a monitorização de utentes em tempo real capaz de agregar dados de diferentes fontes de informação como sensores do dispositivo Android ou outros dispositivos externos capazes de retirar dados.

Este projeto é, também, de grande interesse pessoal, pois oferece uma experiência de programação em *Android*, como também será um projeto com um grande impacto na área de monitorização de saúde que puderá ter um grande impacto e contribuição para a sociedade.

1.3 Objectivos

O projeto MobileVJ visa o desenvolvimento de um sistema de monitorização remoto em tempo real capaz de recolher a informação proveniente de dispositivos externos de recolha de sinais como o VJ, os dados recolhidos pelos sensores incorporados nos dispositivos móveis e a recepção desta informação pelo utilizador que pretende analisar os dados recolhidos pelo sujeito monitorizado.

Com esta dissertação é pretendido criar uma aplicação móvel capaz de permitir a seleção de sensores presentes no dispositivo Android e conseguir agregar a informação recolhida tanto por sensores, como por dispositivos externos. Também é pretendido criar uma aplicação cliente que seja capaz de receber os dados em tempo real, recolhidos pelo utilizador que está a ser monitorizado.

1.4 Estrutura do documento

O presente documento está dividido em 5 capítulos. Desta forma, correspondem aos seguintes capítulos: 1. Introdução; 2. Levantamento de Tecnologias e Estado de Arte; 3. MobileVJ; 4. Resultados e Discussão e por fim, o ponto 5. Conclusão e Trabalhos Futuros.

O primeiro capítulo, relativo à Introduçao, tem como intuito efetuar uma introdução ao tema através da explicação da motivação, como também o enquadramento e os objetivos desta dissertação. O ponto 2 tem como objetivo a demonstração das tecnologias existentes para a criação de um sistema de monitorização em tempo real, como também algumas aplicações com funcionalidades semelhantes. Relativamente ao ponto 3 são apresentados os requisitos, definidos os atores presentes no sistema, como também as suas *user stories*. Foi também descrito de forma pormenorizada a implementação do projeto desta dissertação - o sistema MobileVJ. Quanto ao ponto 4, que é

relativo aos Resultados e Discussão, são descritos alguns dos testes realizados para comprovar o funcionamento do sistema implementado, tendo espaço para retirar algumas conclusões dos resultados obtidos com os testes em questão. Por último, o capítulo 5, tem como objetivo a conclusão de uma forma geral deste projeto, como também apontar algumas ideias de trabalhos futuro no âmbito deste tema de dissertação.

Capítulo 2

Levantamento de Tecnologias e Estado da Arte

Neste capítulo, é realizado um levantamento do estado atual da área em que se enquadra esta dissertação, sendo assim efetuada uma análise inicial ao funcionamento da tecnologia *bluetooth*, tendo em conta que expõe uma funcionalidade como comunicação entre dispositivos via transmissão áudio de maior relevância para o objetivo final deste trabalho.

No subcapítulo seguinte, é revista a tecnologia existente na área da computação móvel aplicada à agregação de sensores via *bluetooth*, sendo mencionado o papel dos *smartphones* na sociedade e a importância da escolha do sistema operativo antes do desenvolvimento de uma aplicação móvel. Subsequentemente, os sensores presentes *smartphones* e os dispositivos vestíveis de monitorização existentes no mercado são analisados.

Finalmente, é efetuada a seleção e levantamento de sistemas atuais que utilizam aplicações móveis para a agregação e recolha de sensores.

2.1 Introdução ao *Bluetooth*

Da mesma forma que os seres humanos comunicam entre si, os *gadgets* também têm formas de comunicar. Como o foco principal desta dissertação é a criação de uma aplicação capaz de agrupar dados de sensores, a tecnologia responsável por esta função será o *Bluetooth*.

Bluetooth é uma tecnologia criada em 1994 e foi concebida como uma alternativa sem fios para cabos de dados através da utilização de transmissoes de rádio para conetar os dispositivos à distancia, e transferir dados entre eles. Esta tecnologia foi criada como um padrão aberto para permitir a conectividade e a colaboração entre diferentes produtos e indústrias [1].

Há biliões de dispositivos de tecnologia *Bluetooth* no mercado atual e na verdade, hoje em dia é difícil encontrar um dispositivo que não tenha esta tecnologia. Por exemplo, já é habitual encontrar o *bluetooth* em *smartphones*, carros, computadores, TVs, auriculares, consolas de jogos, *tablets*, entre outros [2].

Mas com o desenvolvimento do *Buethooth* ao longo dos anos e a sua nova funcionalidade Low Energy, hoje em dia é também possível encontrar esta tecnologia em objetos utilizados na vida quotidiana, tais como papel, roupa, calçado e muito mais [2].

2.1.1 O porquê do Bluetooth

De uma maneira mais formal, é possível dizer que o *Bluetooth* é uma tecnologia de conexão *wireless* (sem fios) que transmite dados de uma forma rápida e com segurança entre vários dispositivos, como telemóveis, *notebooks*, *joysticks* de consolas e muito mais. Contudo, a tecnologia tem sofrido alguns problemas, tais como: baixo alcance, velocidade problemática em alguns casos, gasto altíssimo de energia e riscos de segurança até a versão 3.0. O *Bluetooth* 3.0 trouxe uma velocidade muito maior do que a versão anterior (2.0) ao integrar o padrão de comunicação IEEE 802.11 — utilizado antes apenas para conexões de internet *wireless*. Sendo assim, a tecnologia passou a transmitir em média 24 Mbit/s [3].

Tabela 2.1: Evolução de tecnologia *bluetooth* ao longo das suas versões (Adaptado de [15])

Versão	Taxa de transmissão	Alcance	Retrocompatibilidade	Características especiais
Bluetooth v2.0 + EDR	2.1 Mbit/s	10m	Compatível com todas as versões anteriores	EDR(Enhanced Data Rate) para transferência de dados rápida
Bluetooth v2.1 + EDR	3 Mbit/s	10m	Compatível com todas as versões anteriores	EDR(Enhanced Data Rate) para transferência de dados rápida
Bluetooth v3.0 + HS	24 Mbit/s	12m	Compatível com todas as versões anteriores	HS(Hight Speed)
Bluetooth v4.0 + LE	24 Mbit/s	> 61m	Compatível com todas as versões anteriores	LE(Low Energy) para continua transferência em dispositivos de baixo consumo de energia

Contudo, a versão 4.0 trouxe grandes inovações ao mundo das tecnologias e abriu os horizontes com a possibilidade de criação de novos dispositivos e uso cujas capacidades são limitadas apenas pela imaginação do criador [1].

Como se pode ver na tabela 2.1, a versão 3.0 focou muito mais o aspeto de velocidade — o que também foi importante, já que o uso de *Bluetooth HS* para conexões de alta velocidade se tornou comum — o *Bluetooth* 4.0 focou-se em resolver outros problemas. A economia de energia foi um desses factores [4].

Agora a ferramenta Low Energy torna a tecnologia *Bluetooth* mais inteligente em relação à quantidade de energia que é utilizada, pois a maior parte dela era desperdiçada para pouca transferência. Muitos dos produtos são bastante compactos e não existe necessidade de utilização

de energia extra para transferir pouca informação que permite uma durabilidade de bateria muito maior.

2.2 Computação móvel

A área da computação móvel sofreu um grande crescimento com a evolução do mercado de dispositivos móveis inteligentes também conhecidos por *smartphones* [5]. A computação móvel pode ser representada como um novo paradigma computacional que permite ao usuário desse ambiente ter acesso instantâneo aos serviços independentemente da sua localização. As aplicações desenvolvidas para os *smartphones* podem ser utilizadas em qualquer ponto do mundo, mas não por qualquer um dos dispositivos, pois o seu desenvolvimento depende muito do seu ambiente ou plataforma de utilização. Estas aplicações são feitas com diferentes finalidades e podem usufruir diferentes capacidades de ambiente de de utilização.

2.2.1 Smartphones

Desde a invenção dos telemóveis, no final do século XX, verificou-se alterações enormes no que diz respeito à vida quotidiana das pessoas, o que originou mudanças comportamentais profundas na sociedade, especialmente na maneira das pessoas se relacionarem entre si. Desde que os telemóveis existem, estes sofreram inúmeras alterações no que toca a dimensões e capacidades. A evolução de telemóveis básicos para dispositivos móveis inteligentes (*smartphones*, *tablets, laptops*, etc) veio alterar completamente o padrão da comunicação. Hoje em dia, os telemóveis podem ser considerados como mais do que simples aparelhos de comunicação, visto que para além de terem removido barreiras geográficas, foi principalmente graças a eles que se deu uma grande mudança cultural onde os seus utilizadores são incitados a procurar e adquirir constantemente nova informação e estabelecer novas conexões com conteúdos cada vez mais difundidos [6].

Atualmente, sobretudo em países industrializados, pelo menos cada indivíduo é proprietário de um telemóvel. Em 2015, o número total de subscrições de telemóveis, a nível mundial, era de cerca de 7,085 biliões. No final de 2019, é de prever que o número de subscrições de telemóvel atinja os 9,3 biliões [7, 17].

Apesar da maior parte das subscrições de telemóvel corresponderem a dispositivos básicos, o aumento das subscrições nos últimos anos deve-se sobretudo à aquisição de tecnologia móvel "inteligente" ou *smartphones*. O aparecimento dos dispositivos móveis constitui uma das maiores histórias de sucesso dos últimos anos, uma vez que, num período de tempo relativamente curto, conseguiu-se introduzir de forma significativa na sociedade, atraindo subscritores de todas as idade, desde crianças até aos idosos [6].

"O número de utilizadores de *smartphones* deverá crescer de 1,5 bilhão em 2014 para cerca de 2,5 biliões em 2019, com taxas de penetração de *smartphones* aumentando também. Pouco mais de 36 por cento da população do mundo está projetado para usar um smartphone em 2018, com um aumento desde os 10 por cento em 2011"[8].

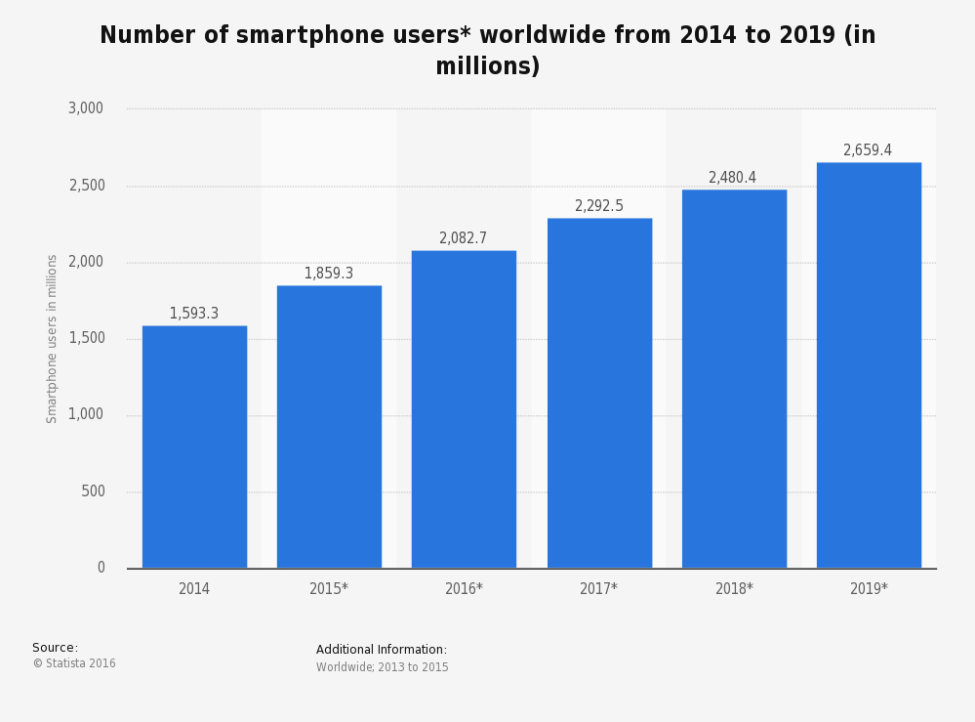

Figura 2.1: Número de utilizadores de *smartphones* desde 2014 a 2019 (Adaptado de [8])

A principal razão desta evolução é o facto do smartphone ser um dispositivo móvel multifuncional, que pode oferecer desde uma simples comunicação até à navegação na internet sendo que os mesmos podem ser considerados os "computadores de bolso" [9].

Estes dispositivos funcionam à base de aplicações móveis, software desenvolvido propositadamente para um ambiente como Android, iOS ou WindowsPhone entre muitos outros, com a sua funcionalidade mais específica devido aos recursos limitados que os dispositivos móveis oferecem em comparação com os portáteis [11]. As aplicações tendem a utilizar diferentes recursos que um smartphone pode oferecer, tais como a câmara, microfone, acelerómetro de 3 eixos, giroscópio de 3 eixos, sensores de proximidade, sensor de luz ambiente, sensor tátil, magnetómetro e GPS [10].

Resumindo, o smartphone é um dispositivo muito compacto e multifuncional que acaba por ser integrado no quotidiano dos seres humanos apesar de demonstrar alguma desvantagem em relação aos computadores no processamento e armazenamento de dados.

2.2.2 Sistema Operativo

As aplicações móveis dos *smartphones* são desenvolvidas para correrem num sistema operativo próprio do dispositivo, pelo que a primeira fase de planeamento da aplicação passa pela seleção desse mesmo sistema operativo. Atualmente, no panorama das tecnologias móveis há um leque de pouco mais de cinco sistemas operativos, havendo dois que se destacam: o Android e o iOS. No entanto, no que toca à percentagem de vendas de dispositivos móveis, o Android tem destronado a concorrência como é possível verificar no gráfico da Figura 2.5 [12].

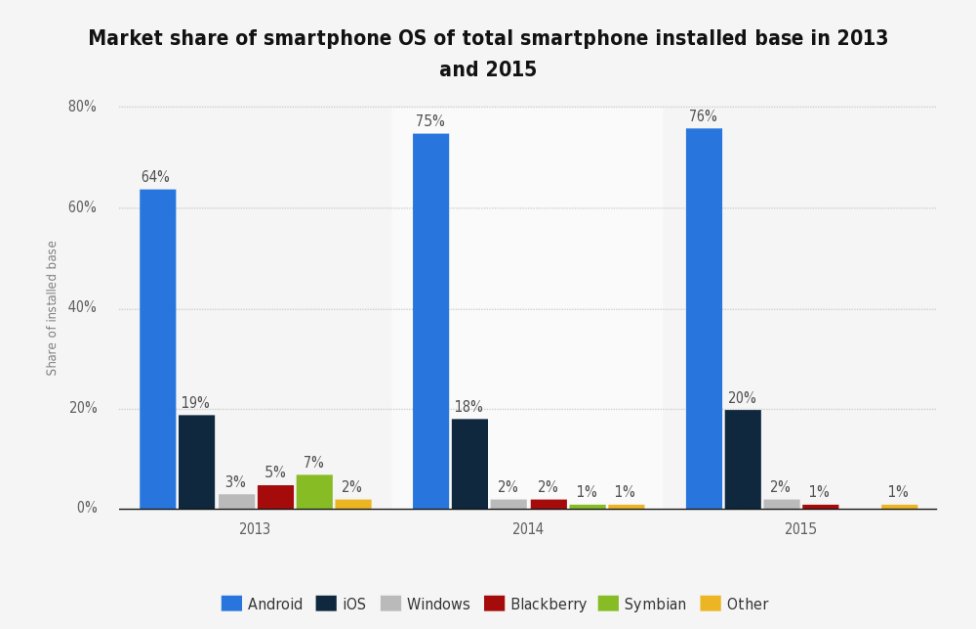

Figura 2.2: Percentagem de mercado de sistemas operativos móveis nos últimos 3 anos (Adaptado de [12])

Atualmente, este sistema operativo suporta mais de metade dos *smartphones* em todo o mundo [12], o que levou a que também fosse escolhido como a plataforma ideal para o desenvolvimento do módulo agregador e da aplicação móvel apresentada nesta dissertação. A versão mínima do sistema operativo para a aplicação a desenvolver será Android 4.3 (nível API 18), pois oferece o suporte à plataforma integrada para Bluetooth Low Energy que terá um papel importante nesta dissertação [13].

2.2.3 Sensores presentes em *smartphones*

Os telemóveis, nos dias de hoje, fornecem uma maior funcionalidade do que apenas uma chamada de voz e uma mensagem de texto. Diferentes sensores integrados nos *smartphones* permitem tirar grandes proveitos destes pequenos dispositivos, como por exemplo saber a localização exata através de satélites GPS ou pode ser derivada de pontos de acesso de informação *Wi-Fi* ou da localização de torres de comunicação móvel.

O sistema operativo Android disponibiliza manipulação de diferentes sensores a nível de hardware e software. Os sensores baseados em hardware são componentes físicos incorporados no dispositivo do smartphone. Os sensores hardware obtêm os seus dados medindo diretamente as propriedades ambientais específicas, tais como a aceleração, a força do campo magnético ou alteração angular. Sensores baseados em software não são dispositivos físicos, embora eles imitam sensores baseados em hardware. Os sensores baseados em software derivam dos dados a partir de um ou mais sensores baseados em hardware e são por vezes chamados sensores virtuais ou sensores sintéticos. O sensor de aceleração linear e o sensor de gravidade são exemplos de sensores

com base em software [14].

Estes sensores podem ser de três categorias: sensores de movimento, sensores ambientais e sensores de posição. Os sensores de movimento medem forças de aceleração e rotação ao longo de três eixos. Esta categoria inclui acelerómetros, sensores de gravidade, giroscópios e sensores de rotação do vetor. Os sensores ambientais medem vários parâmetros ambientais, tais como temperatura ambiente e a pressão do ar, a iluminação e a humidade. Esta categoria inclui barómetros, fotómetros e termómetros. Por fim, temos os sensores de posição que medem a posição física de um dispositivo. Esta categoria inclui sensores de orientação e magnetómetros [14]. Os sensores oferecidos pela *framework* Android estão demonstrados na Figura 2.3.

Sensores de Movimento		Sensores Ambientais		Sensores de Posição	
Acelerómetro	*Hardware*	Temperatura de ambiente	*Hardware*	Magnetómetro	*Hardware*
Giroscópio	*Hardware*	Humidade relativa do ar	*Hardware*	Orientação	*Software*
Gravidade	*Software / Hardware*	Sensor de pressão atmosférica	*Hardware*	Proximidade	*Hardware*
Vvetor rotacional	*Software / Hardware*	Luminosidade	*Hardware*		
Aceleração linear	*Software / Hardware*				

Figura 2.3: Conjunto de sensores presente em *smartphones* no *SO Android*, divididos de acordo com a sua função(Adaptado de [14])

2.3 Sensores vestíveis de ECG

Os sensores vestíveis ou as *Wearables* são os dispositivos tecnológicos que podem ser utilizados pelos usuários como peças do vestuário. Estes são feitas para usufruir da tecnologia a partir de uma perspectiva mais cômoda e natural para os usuários, faz parte do processo de evolução da onipresença da informática e computação na vida das pessoas. Os dispositivos wearables são um passo para a concretização da IoT, que se caracteriza por manter a constante conectividade entre diferentes tipos de objetos comuns no cotidiano dos indivíduos. Por exemplo, os óculos, relógios, pulseiras ou mesmo as camisas, estes são alguns dos exemplos de como a tecnologia móvel pode ser inserida em diferentes acessórios, seja como uma fonte de informação ou comunicação para os seus utilizadores [29] .

Nesta secção serão descritos alguns dos *wearables* que serão potencialmente introduzidos no projeto MobileVJ existentes no mercado capazes de providenciar uma monitorização remota do sinal de eletrocardiograma sem custos elevados sendo também possível emparelha-los com *smartphones*.

Grande parte das aplicações móveis que se focam na monitorização da atividade cardíaca recorrem a dispositivos externos capazes de recolher o eletrocardiograma do utilizador. “Apesar dos sensores de ECG já existirem há bastantes anos, estavam até há relativamente pouco tempo associados a máquinas de grande porte, com muitos fios e extremamente dispendiosas para poderem ser

utilizadas fora do ambiente hospitalar. Contudo, ultimamente com os avanços tecnológicos este tipo de dispositivos têm-se tornado cada vez mais acessíveis ao público através de sistemas mais baratos, discretos, vestíveis e prontos a utilizar" [16].

2.3.1 Zephyr HxM® BT

Zephyr HxM® BT (Figura 2.4) é um monitor de frequência cardíaca, velocidade, distância percorrida e nível de intensidade que utiliza transmissão via Bluetooth capaz de transmitir o sinal até 10 metros e funciona com as versões de Android 4.3 ou superior[17, 18].

O aparelho é alimentado por uma bateria recarregável de polímero de lítio que dura 26 horas. Ele vem com um carregador USB e leva 3 horas para recarregar completamente a bateria. A Zypher também oferece *Software Development Kit* (HxM BT SDK) para o desenvolvimento de aplicações de plataforma Android [17].

Figura 2.4: Dispositivo Zephry HxM® (Adaptado de [18])

2.3.2 Vital Jacket®

O VitalJacket®(Figura 2.5) é um dispositivo médico que conjuga a tecnologia têxtil com soluções avançadas de engenharia biomédica, que permite uma monitorização contínua durante 72 horas seguidas. O dispositivo permite a configuração para adquirir diferentes sinais vitais tais como ECG, temperatura, respiração, postura, saturação de oxigênio, etc. Esta t-shirt vem equipada com um registador de dados e Bluetooth que permite a transferência de dados em tempo real para os dispositivos móveis [19].

Figura 2.5: Kit VitalJacket® (Adaptado de [19])

2.3.3 Under Armour E39

Empresa Under Armour e Zephyr Technology desenvolveram uma cinta para a monitorização de atletas, a Under Armour E39 (Figura 2.6). Esta é uma cinta com sensores integrados capaz de realizar a monitorização de frequência cardíaca, frequência respiratória, a temperatura do corpo humano e a sua aceleração com uma capacidade de 2 gigabytes de armazenamento [30].

Figura 2.6: A cinta E39

2.3.4 hWearT

A camisola hWearTM (Figura 2.7) desenvolvida pela empresa Health Watch Technologies LTD, permite realizar a monitorização de batimento cardíaco, pressão sanguínea e deteção de irregularidades cardíacas de forma remota sem qualquer incómodo para o utilizador. Este dispositivo em conjunto com o leitor HealthWatch's MasterCaution, permite o envio de alertas em *realtime* não só para os médicos como também para os pacientes em caso de arritmias, isquemias, anormalidades respiratórias e imobilidade prolongada [31].

Figura 2.7: A cinta E39

2.4 Sistemas móveis de monitorização de saúde

No Google Play existe um grande número de aplicações para dispositivos Android que usufruem de funcionalidades de Bluetooth e sensores incorporados dos smartphones. Porém, efetuando pesquisas não foi possível encontrar aplicações de monitorização de saúde para a recolha de dados dos dispositivos externos, tanto como dos dispositivo Android.

Desta forma, nesta secção serão explorados alguns sistemas que tratam de agregação de dados através dos sinais Bluetooth ou sensores internos dos dispositivos móveis. Desta forma, nesta secção serão explorados alguns sistemas de monitorização de saúde em tempo real

2.4.1 Science Journal

A Science Journal é uma aplicação Google (Figura 2.8). De um modo geral esta ferramenta possibilita efetuar ciência com um smartphone. Permite experimentar e interagir com o ambiente usando diferentes sensores presentes no telemóvel com sistema Android, bem como sensores externos compatíveis com o dispositivo móvel. Esta aplicação mede e grava os dados em tempo real, e depois converte os dados em gráficos e tabelas de fácil leitura. Os utilizadores podem armazenar vários projetos na aplicação e usar ferramentas como um acelerómetro e microfone, além de medidores de som e luz, de modo a reunir informações para os seus projetos. Por exemplo, um utilizador pode gravar o seu ritmo de corrida todos os dias durante uma semana, e, em seguida, traçar o seu progresso num gráfico de linhas.

Ou seja, esta aplicação usa sensores para representar graficamente dados, registar as experiências do utilizador, como também organizar ideias, projetos e dúvidas.

Embora o número de sensores disponíveis para uso ainda seja pequeno, o Google afirma que futuramente irá trabalhar com especialistas da comunidade científica de modo a melhorar continuamente esta aplicação.

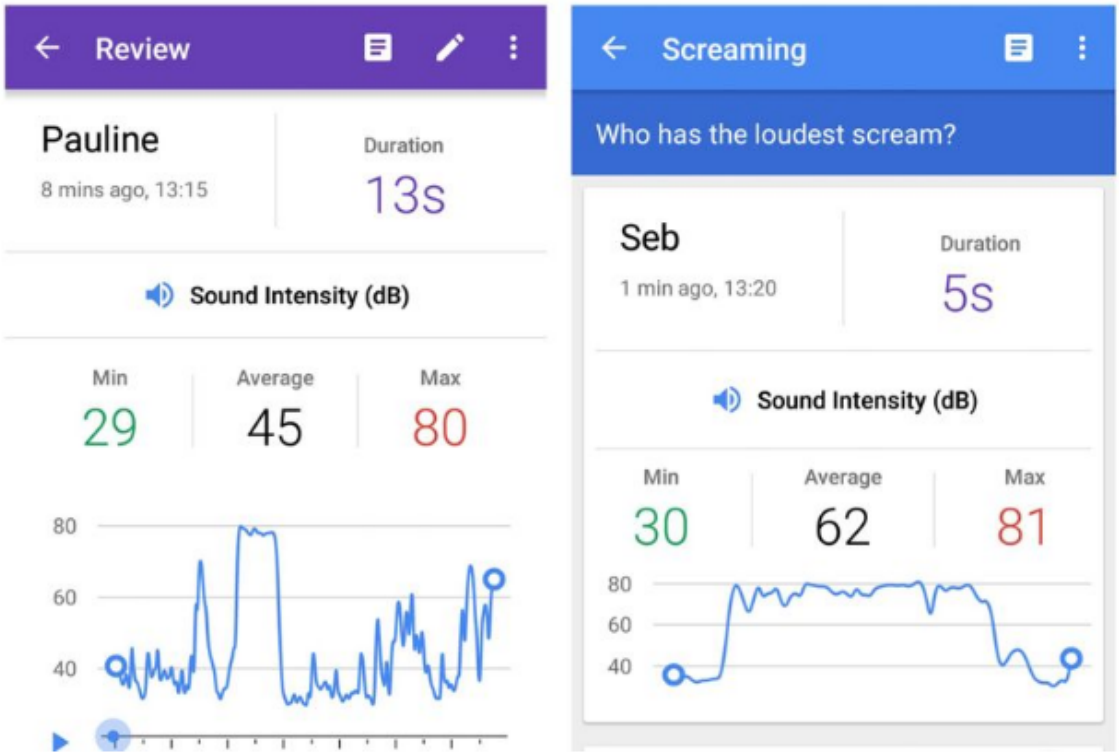

Figura 2.8: Interface da aplicação Science Journal

2.4.2 A Real-Time Health Monitoring System for Remote Cardiac Patients Using Smartphone and Wearable Sensors

A Real-Time Health Monitoring System for Remote Cardiac Patients Using Smartphone and Wearable Sensors é um sistema de monitorização concebido para fornecer uma interface entre o médico e os pacientes para a comunicação de duas vias [26]. Este sistema utiliza o dispositivo Android para o sujeito monitorizado com um dispositivo Zephyr HxM e uma interface Web para o médico. Este sistema tem a capacidade de gerar alertas de emergência com base na comparação dos valores obtidos pela medição do paciente com os valores pré-definidos, de forma a informar o médico caso haja algum cuidado a tomar. Este sistema oferece a monitorização de pacientes com um atraso entre 4 a 6 minutos [26].

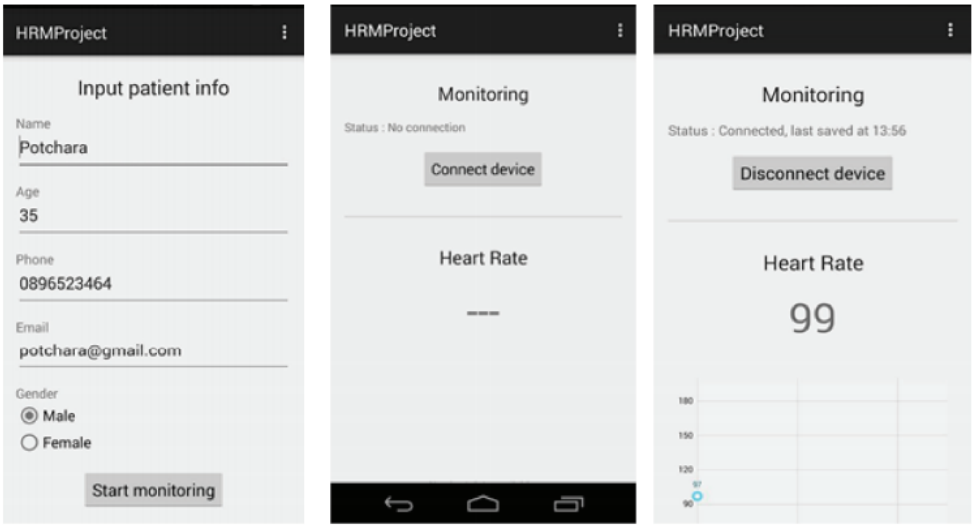

Figura 2.9: Interface da aplicação: informações básicas do paciente, notificação de conexão e monitorização em tempo real

2.4.3 eCAALYX

A aplicação Enhanced Complete Ambient Assisted Living Experiment (eCAALYX) pretende desenvolver um sistema remoto de monitorização de pessoas idosas que padeçam de doenças crónicas de forma a melhorar a qualidade de vida, promover a autonomia e ao mesmo tempo reduzir os custos hospitalares [27].

Este projeto pertencente à União Europeia faz parte da AAL JP (The Active and Assisted Living Joint Programme) e recorre a uma aplicação móvel para servir de interface com os idosos e possui uma arquitetura (Figura 2.10) que lhe atribui as seguintes funções:

1. Intervir com um intermediário perfeito entre os sensores de sinais vitais vestíveis usados pelas pessoas idosas e o site da internet dos profissionais da área da saúde;

2. Reportar os alertas/avisos e dados obtidos através dos sensores e da localização geográfica do utilizador (a partir do GPS do smartphone);

3. Efetuar o processamento de dados obtidos pelo sensor para identificar informações de nível superior, incluindo irregularidades simples de detetar como taquicardia e sinais de infeções respiratórias, com base no conhecimento médico programado;

4. Permitir a acessibilidade da interface do utilizador para que o utilizador consiga avaliar os detalhes médicos mais recentes recebidos a partir dos sensores, efetuar novas medições e comunicar com os profissionais de saúde.

Os objetivos devem ser alcançados não só por melhorar a comunicação com seus cuidadores, mas também permitir que os idosos monitorizem a sua saúde regularmente e educá-los sobre como evitar situações de risco, entre outros.

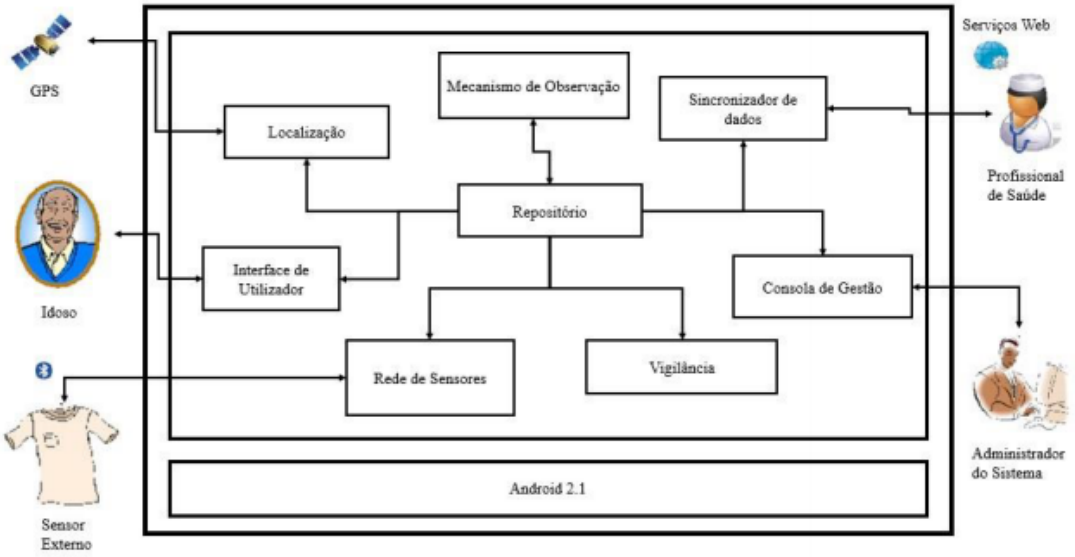

Figura 2.10: Arquitetura da plataforma móvel eCAALYX

2.4.4 An Ultra-Wearable, Wireless, Low Power ECG Monitoring System

Este é um sistema de monitorização de ECG vestível sem fios e de baixo consumo de energia [28]. Este projeto possui 4 grande objetivos:

Ultra-Wearability – este sistema faz uso do QUASAR's, um sensor inovador e uma ligação de sensores sem fios ultracompacto especialmente desenhado para aplicações portáteis;

High Throughput – este projeto tem também o objetivo de alcançar uma alta taxa de transferência de rede, que é necessário para uma baixa latência do sistema de monitorização;

Low Power - O baixo consumo de energia contribui não só para o prolongamento do tempo de vida, mas também para a miniaturização do sistema. Desta forma, este sistema possui um transcetor de energia muito baixo que consome menos de 10 mA em modo de transmissão (1 Mbps, 0 dBm) e 22 mA no modo de receção;

Universal Connectivity – este sistema foi concebido para ser capaz de transmitir dados pelas mais comuns interfaces de comunicação, incluindo USB, Ethernet e Wi-Fi.

O sistema de monitorização de ECG pode ser funcionalmente dividido em quatro subsistemas: sensores de ECG, dados de amostragem, transmissão sem fios e host interface. Neste projeto foram propostas três diferentes arquiteturas de sistema (Figura 2.11).

De uma forma resumida, pode-se constatar que a primeira arquitetura [Figura 2.12(a)] consiste em múltiplos sensores de ECG, um módulo de amostragem de dados, um nó de ECO e uma estação de base. A segunda arquitetura usa o ECO tanto para as amostragens de dados como também para a transmissões sem fio. Todos os sinais de ECG são alimentados diretamente para os canais ADC na ligação ECO. A terceira arquitetura [Figura 2.12(c)] usa uma ligação de ECO para cada sensor de ECG. Esta arquitetura é parecida com a segunda relativamente ao facto da ligação de ECO realizar as funções de amostragem e de comunicação, exceto que uma ligação de ECO serve apenas um sensor de ECG neste caso.

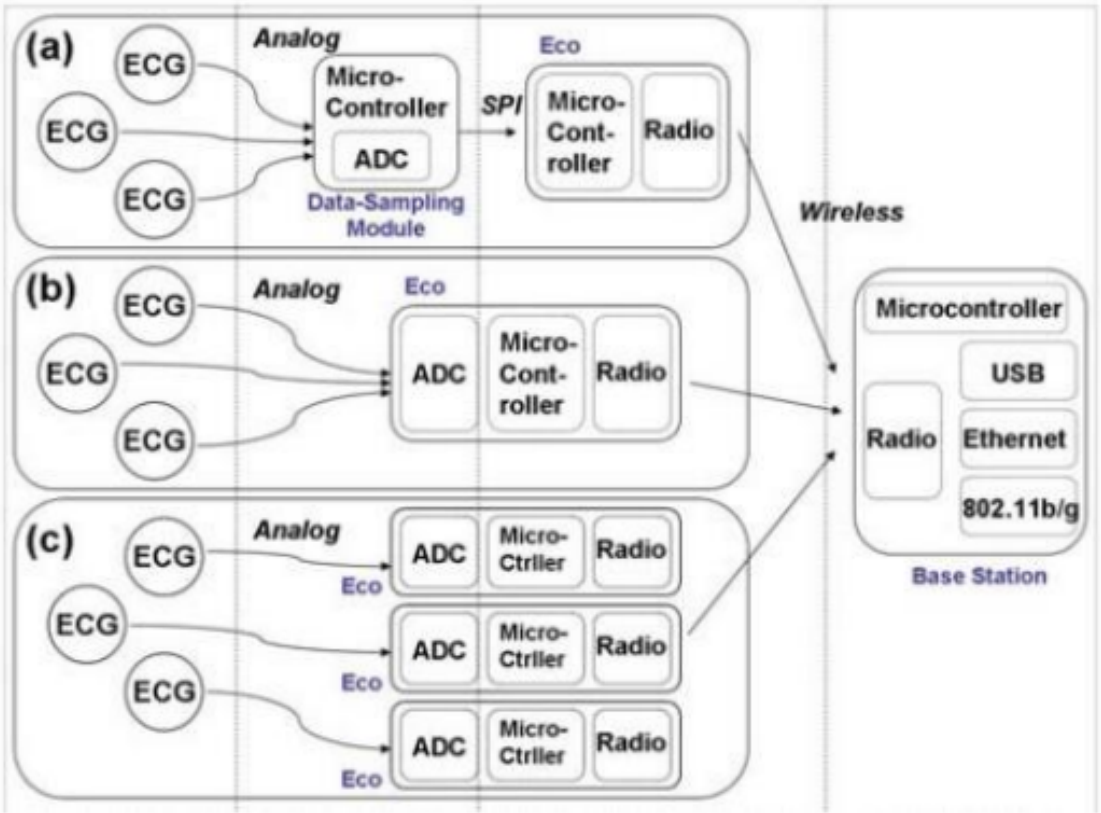

Figura 2.11: Arquitetura da plataforma IEEE Xplore

2.5 Message Brokers

Um Message Broker é um gestor de mensagens, responsável pela troca de mensagens entre o remetente e o receptor. Ao receber uma mensagem, irá verificar quais os seus destinatários e o formato que os mesmos aceitam. Caso o formato seja diferente, este efetuará a operação de conversação necessária, de forma a reduzir a carga sobre os recetores [24].

Os Message Brokers, normalmente, aceitam dois tipos de modelos de partilha de mensagens: Point-to-Point e Publish/Subscribe. Ambos os modelos assentam na transmissão de mensagens visto que o modo como se difundem e são consumidas pode ser um pouco diferente [24].

No módulo **Point-to-Point** os produtores das mensagens enviam os dados para uma fila de espera à sua escolha, sendo que os clientes que têm como o objetivo receber as mensagens da fila escolhida, ficam à espera até que alguma informação seja entregue. Neste modelo podem existir vários produtores a enviar para uma única fila, como também mais de que um consumidor a escutar essa mesma fila. Deve-se ter em conta que neste modelo uma mensagem é entregue apenas a um consumidor, e posteriormente essa mensagem será eliminada da fila para evitar sobrecargas [24].

Uma vez que em Point-to-Point apenas o primeiro cliente recebe a informação desejada, este é também responsável por realizar as operações solicitadas, visto que existe a possibilidade de ser sempre o mesmo cliente a produzir maior quantidade do trabalho, pois não existe um controlo sobre as ordens de leitura da informação [24].

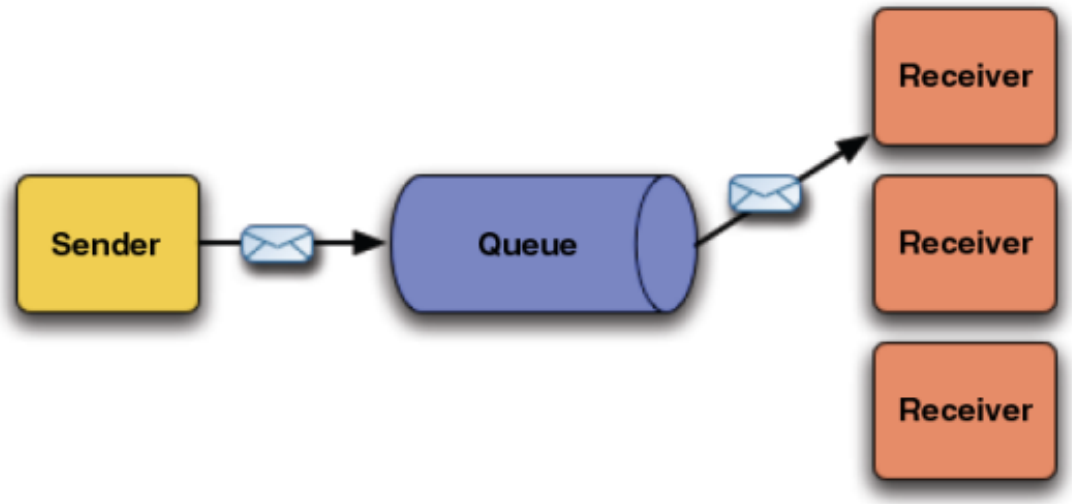

Figura 2.12: Arquitetura Point-to-Point

No módulo **Publish/Subscribe** cada elemento de um sistema distribuído pode ter vários papéis: Publisher; Subscriber; Publisher/Subscriber. Os Publishers enviam as mensagens para todos os clientes subscritos, os Subscribers. Enquanto os Subscribers definem os seus interesses, de forma a subscrever um amplo número de Publishers [24].

Publishers não têm a informação quanto aos clientes que ficam à espera da entrega das suas mensagens. Estes definem os formatos nas suas mensagens, para que os gestores intermediários possam efetuar o reencaminhamento para todas os clientes subscritos a este formato das mensagens. No caso dos Message Brokers, estes são os responsáveis pela gestão e reencaminhamento das mensagens [24].

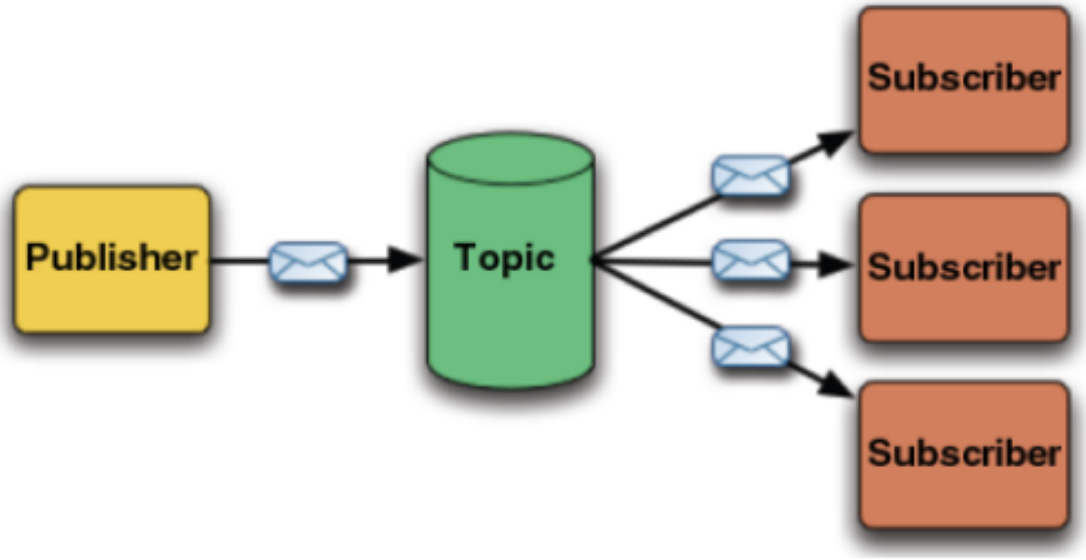

Figura 2.13: Arquitetura Publish/Subscribe

2.5.1 RabbitMQ

RabbitMQ, da Pivotal Software, é um message broker que permite enviar mensagens através de um processo chamado de producer/produtor e posteriormente ler as mensagens na fila de espera através de um objeto consumer/consumidor. É disponibilizado um servidor para instalação em máquinas com os diversos sistemas operativos (Windows, Debian, Ubuntu, Fedora, entre outros). Os seus clientes podem ser desenvolvidos nas variadíssimas linguagens de programação, designando-se por uma framework poliglota. Tem suporte para os principais modelos de partilha como Point-to-Point e Publish/Subscribe, assim como vários modelos de subscrição [25]

Neste processo de envio e leitura de mensagens encontram-se 3 conceitos fundamentais:

1. **Exchange** – São pontos de entrada para todas as mensagens publicadas pelos produtores. As mensagens nunca poderão ser enviadas para uma fila diretamente sem passarem por um Exchange.

2. **Queue** – Fila onde são armazenadas as mensagens.

3. **Binding** – Bindings são conjuntos de regras para efetuar a ligação entre as queues e as exchanges. Neste processo define-se para que queue deve ser enviada a mensagem recebida no ponto de entrada.

As mensagens nas filas são armazenadas de forma independente. Cada mensagem contém a informação necessária ao seu processamento. Uma mensagem pode ser um objeto serializado, um email a enviar, uma imagem, ou outro elemento qualquer.

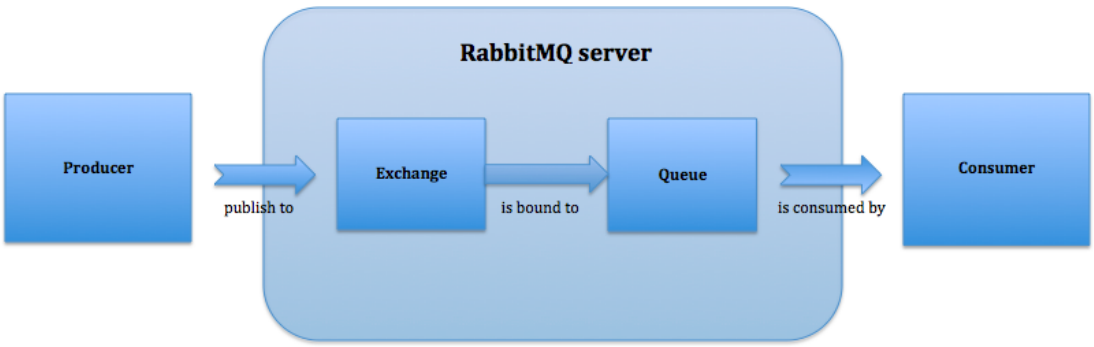

Figura 2.14: Representação do processamento das mensagens

Com a utilização da aplicação RabbitMQ é possível destacar algumas vantagens, tais como:

1. **Scalability**: Um consumidor lê uma mensagem da fila, mas não tem que existir necessariamente apenas um consumidor. Por isso é possível criar múltiplos consumidores.

2. **Clustering**: Múltiplos servidores RabbitMQ na rede local podem ser agrupados de forma a criarem um único broker lógico.

3. **Highly Available Queues**: As filas podem ser clonadas para múltiplas máquinas num cluster, garantindo que, mesmo em caso de falha de hardware as mensagens são guardadas.

4. **More responsive applications**: através da delegação das tarefas/processamento, a aplicação RabbitMQ foca-se em realizar as tarefas realmente importantes para o utilizador ou para o serviço.

2.5.2 Apache Kafka

O Apache Kafka é um opensource com envio distribuído de mensagens que permite que o utilizador crie aplicações em realtime usando dados de streaming. O sujeito pode enviar dados de streaming, como sequências de cliques de sites, transações financeiras e logs de aplicação, para o cluster do Kafka. Ele reserva os dados e os distribui para aplicativos de processamento de streams incorporados às estruturas, como Apache Spark Streaming, Apache Storm ou Apache Samza [33].

O Kafka foi primeiramente desenvolvido com o intuito de solucionar problemas de entrega de enormes volumes de dados num sistema Publish/Subscribe. Pode ser visto como um commit-log para toda a infraestrutura de um sistema que providencia uma abstração ao nível do mecanismo de extração de mensagens, permitindo tanto subscritores como produtores lerem as mesmas a ritmo arbitrário. A comunicação entre clientes e servidores é bastante simples através do protocolo TCP [33].

O Kafka está a ganhar cada vez mais popularidade em Big Data pois além de ser um projeto open-source de alta qualidade, possui a capacidade de lidar com fluxos de alta velocidade de dados, característica cada vez mais procurada para uso na IoT, por exemplo [32].

Algumas das empresas que usam Kafka são: LinkedIn, Netflix, PayPal, Spotify, Uber, AirBnB, Cisco, Goldman Sachs e SalesForce.

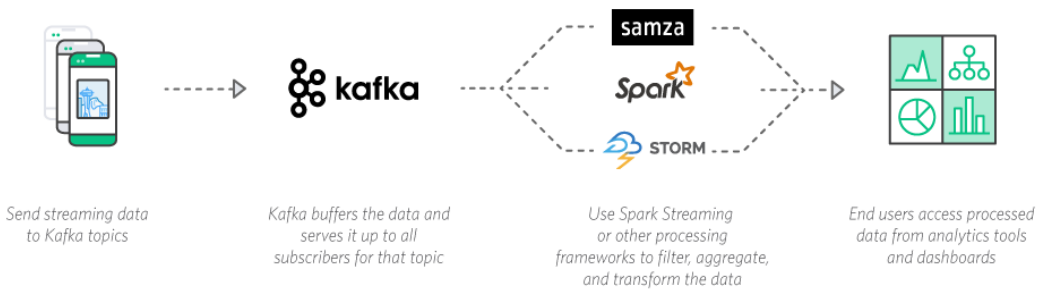

Figura 2.15: Representação do processamento das mensagens

2.5.3 ZeroMQ

ZeroMQ é uma biblioteca de mensagens de alto desempenho desenvolvido pelo iMatrix Corporation, que fornece sockets, as quais transportam mensagens através de vários meios, como por exempo o TCP ou multicast e permite projetar um complexo sistema de comunicação sem um grande esforço. Este sistema tem suporte para os principais modelos de partilha como Point-to-Point e Publish/Subscribe [35].

Ao início este sistema foi introduzido como um "messaging middleware", entretanto mudou-se para "TCP on steroids" e hoje em dia é considerado um "new layer on the networking stack".

O seu objetivo principal é garantir rapidez mesmo em redes bastante complexas, sendo que a sua ideologia passa pela implementação de latência zero, de administração zero, de custo zero e de desperdício zero.

O sistema ZeroMQ é considerado bastante rápido. Ele é ordens de grandeza mais rápida do que a maioria dos sistemas de mensagens AMQP e pode obter este elevado desempenho por causa das seguintes técnicas:

- Pode fazer uso de transportes eficientes, tais como Multicast ou Y-suite IPC transport;
- Ele faz uso de lotes de mensagens inteligentes. Isto permite que o zeroMQ utilize de forma eficiente uma conexão TCP / IP, minimizando não só a sobrecarga de protocolo, mas também as chamadas do sistema.

2.6 Conclusão

Visto a necessidade de um sistema de monitorização em tempo real que tenha em conta os fatores externos, foram demonstradas algumas das tecnologias wearable capazes de executar a monitorização de saúde sem algum incómodo para o utilizador final, também como a evolução das tecnologias móveis e os seus sensores incorporados, capazes de recolher uma informação adicional que possa ser útil para uma análise mais profunda.

A tecnologia Message Brockers será testada para a monitorização em tempo real de saúde, de forma a conseguir deminuir o consumo de recursos dos dispositivos móveis. Para esta dissertação foi escolhida tecnologia RabbitMQ que permite uma gestão fácil de mensagens, e um controlo símples de tráfego através da página de gestão que esta oferece.

Capítulo 3

MobileVJ

O objetivo desta dissertação envolve o desenvolvimento de um sistema capaz de sincronizar a comunicação entre os clientes de tipo publisher e subscriber através de um servidor, de forma a recolher os dados pelos sensores presentes num smartphone e pelos sensores presentes em dispositivos externos ligados, sendo possível fazer uma monitorização de dados recolh idos de um ou vários indivíduos a serem monitorizados. Neste capítulo, serão expostos os requisitos necessários ao desenvolvimento do sistema mobileVJ, e a tecnologia a usar. Posteriormente, é apresentado o fluxo de atividades constituintes na aplicação móvel e os protótipos exploratórios dos ecrãs da aplicação de acordo com o fluxo das atividades da aplicação Android e da API a desenvolver. Será apresentada, então, a arquitetura do módulo agregador de sensores de smartphone e de dispositivos externos. Por fim, serão expostas as implementações da aplicação móvel e a aplicação para o servidor e o cliente de tipo subscriber, tendo em conta a necessidade de comprovar a funcionalidade do sistema proposto.

3.1 Análise de requisitos

Para o desenvolvimento do sistema MobileVJ é fundamental conhecer os requisitos que devem ser cumpridos pelo sistema. A análise de requisitos é a primeira fase de desenvolvimento de software quando ficam definidas as funcionalidades do sistema que será desenvolvido.

Assim, recorreu-se à ferramenta de UML Visual Paradigm para desenvolver um diagrama de Casos de Uso que é exposto na Figura 3.1, onde é possível identificar os requisitos fundamentais para o desenvolvimento do módulo agregador e da sua aplicação de demonstração (DemoApp).

Relativamente à aplicação mobile, esta deverá ser capaz de permitir a seleção de sensores existentes no smartphone utilizado, como também permitir a ligação com dispositivos externos e recolher a sua informação, de forma a recolher os dados dos vários sensores em simultâneo e identificar o timestamp e o dispositivo de onde a informação foi recolhida. Para esta dissertação será utilizado o VJ como dispositivo externo oferecido pela empresa Biodevices.

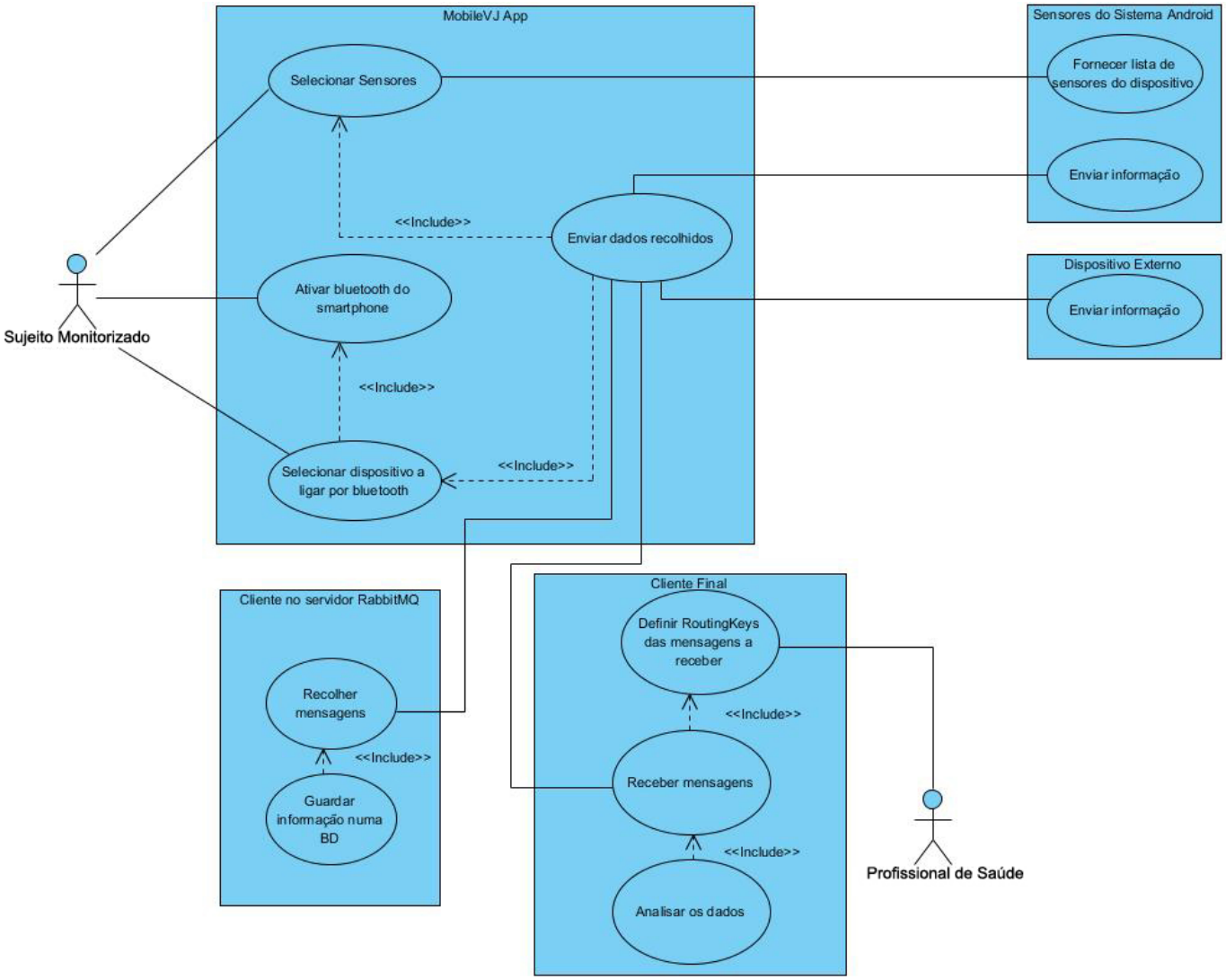

Figura 3.1: Diagrama de Casos de Uso do sistema MobileVJ

O próximo requisito será a monitorização dos dados recolhidos pelos sensores e dispositivos externos. Visto que a dissertação se enquadra no desenvolvimento de um sistema de monitorização para bombeiros e militares, foi então decidido que a monitorização será efetuada através de outros utilizadores do sistema: os analistas. Uma vez que não existe grande importância no próprio bombeiro ou militar analisar os seus próprios dados durante a missão.

Deste modo, foi criada a comunicação entre os utilizadores em monitorização e os analistas através de um sistema de monitorização, utilizando os publishers e subscribers. A Mobile App passou a implementar o sistema publisher e o cliente passou a implementar o sistema subscriber. O servidor passa não só a distribuir as mensagens, mas também a guardar toda a informação numa base de dados noSQL.

O requisito seguinte é a recepção de valores recolhidos pelos diversos utilizadores em monitorização que apenas interessam aos analistas. Isto significa que cada analista pode analisar os dados

de todos os utilizadores, mas apenas de um tipo específico, tais como ECG, temperatura corporal, entre outros.

3.2 Atores

O sistema MobileVJ deve permitir três tipos de utilizadores do sistema que podem ser vistos na Figura 3.2.

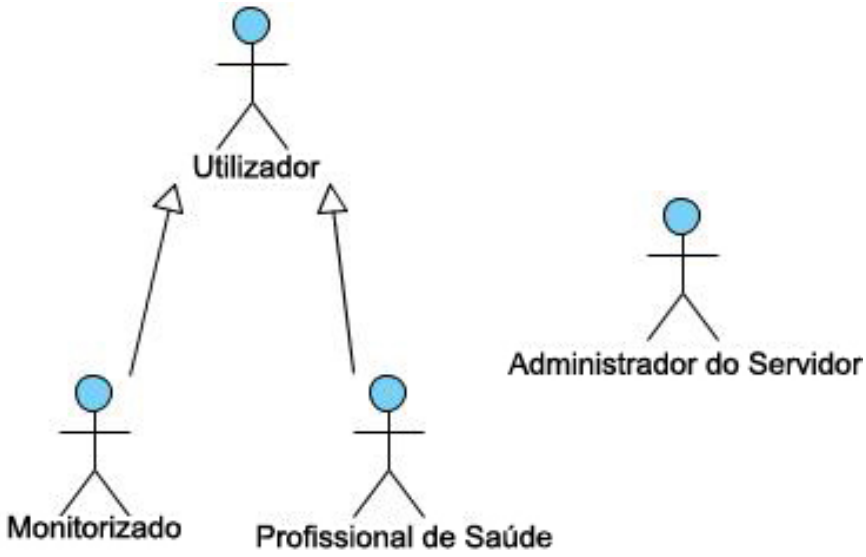

Figura 3.2: Atores

O primeiro utilizador é o sujeito monitorizado. Este será o utilizador da mobile app cujos dados serão agregados pela aplicação e enviados em tempo real para o servidor onde estão a correr os serviços de RabbitMQ.

O segundo tipo de utilizador é o administrador do servidor RabbitMQ. Este pode criar e remover os utilizadores publisher e subscriber, oferecer, alterar e remover as permissões dos serviços de RabbitMQ. O administrador também pode apagar as exchanges e as queues criadas pelos utilizadores. É o único tipo de utilizador que tem acesso a todo o conteúdo informativo que passa pelo servidor e é guardado na base de dados.

O terceiro utilizador é o professional de saúde. É o utilizador que estará subscrito a vários sensores que recolhem dados que são posteriormente enviados para o servidor. A função final deste tipo de utilizador será analisar os dados recebidos em tempo real.

3.3 User Stories

No sistema informático é importante identificar o que cada um dos atores pode realizar no sistema, para que não ocorram inconsistências no mesmo. Para o sistema MobileVJ identificaram-se como relevantes as seguintes user stories que se apresentam na Tabela 3.1.

Tabela 3.1: Tabela de user stories associada ao 3 tipos de utilziadores

Identificador	**Nome**	**Prioridade**	**Descrição**
US001	Ligar Bluetooth	média	Como utilizador monitorizado pretendo ligar bluetooth caso este não se encontrar ligado
US002	Selecionar Sensores	alta	Como utilizador monitorizado pretendo selecionar sensores do dispositivo android
US003	Selecionar Dispositivos	alta	Como utilizador monitorizado pretendo selecionar dispositivos externos a ligar por bluetooth
US004	Iniciar Record	alta	Como utilizador monitorizado pretendo inicar a transmição de dados recolhidos
US005	Parar Record	média	Como utilizador monitorizado pretendo parar a transmição de dados recolhidos
US101	Criar User	alta	Como administrador quero criar utilizador rabbitMQ
US102	Oferecer Permissões	alta	Como administrador quero dar permissões aos utilizadores criados
US103	Alterar Permissões	alta	Como administrador quero alterar permissões do utilizador rabbitMQ
US104	Remover Permissões	alta	Como administrador quero remover permissões do utilizador rabbitMQ
US104	Remover Utilizador	alta	Como administrador quero remover utilizador rabbitMQ
US105	Remover Exchange	média	Como administrador quero remover exchange do servidor
US106	Remover Queue	média	Como administrador quero remover queue existente
US201	Escolher RoutingKey	média	Como profissional de saúde quero escolher um ou vários routing key
US202	Recepção DeDados	alta	Como profissional de saúde quero receber a informação desejada
US203	Escolha de Tipo de Cliente	média	Como profissional de saúde quero escolher a opção de tipo de cliente

3.4 Fluxo de Atividades e Protótipos Exploratórios

Uma vez definidos os requisitos e user stories, partiu-se para o planeamento da forma como o utilizador vai interagir com a aplicação publisher e aplicação subscriber. Este passo é de grande importância na área da computação pois a capacidade de resposta da aplicação é essencial para o sucesso da mesma e os utilizadores deste tipo de serviços esperam resultados imediatos às suas ações. Desta forma, procedeu-se à projeção da aplicação através elaboração de Diagramas de Atividades no programa Visual Paradigm, com o intuito de estabelecer o fluxo viável entre as diferentes atividades.

A mobile app foi então projetada para que ao ser acedida, apareça um Menu Inicial (Figura 3.3 a) com um botão para iniciar a atividade principal de recolha de dados. Também existem duas opções no Action overflow menu da App Bar, a primeira opção "FindVJDevice", permite a escolha do dispositivo externo a ligar por bluetooth (por defeito o dispositivo a ser utilizado será o VJ), a segunda opção "selectSensors", permite a seleção de sensores que o dispositivo móvel nos oferece (Figura 3.3 b). Caso o bluetooth do dispositivo se encontre desligado, a aplicação sugere ao utilizador ligar o bluetooth. Este menu corresponde às user stories US001, US002, US003 e US004.

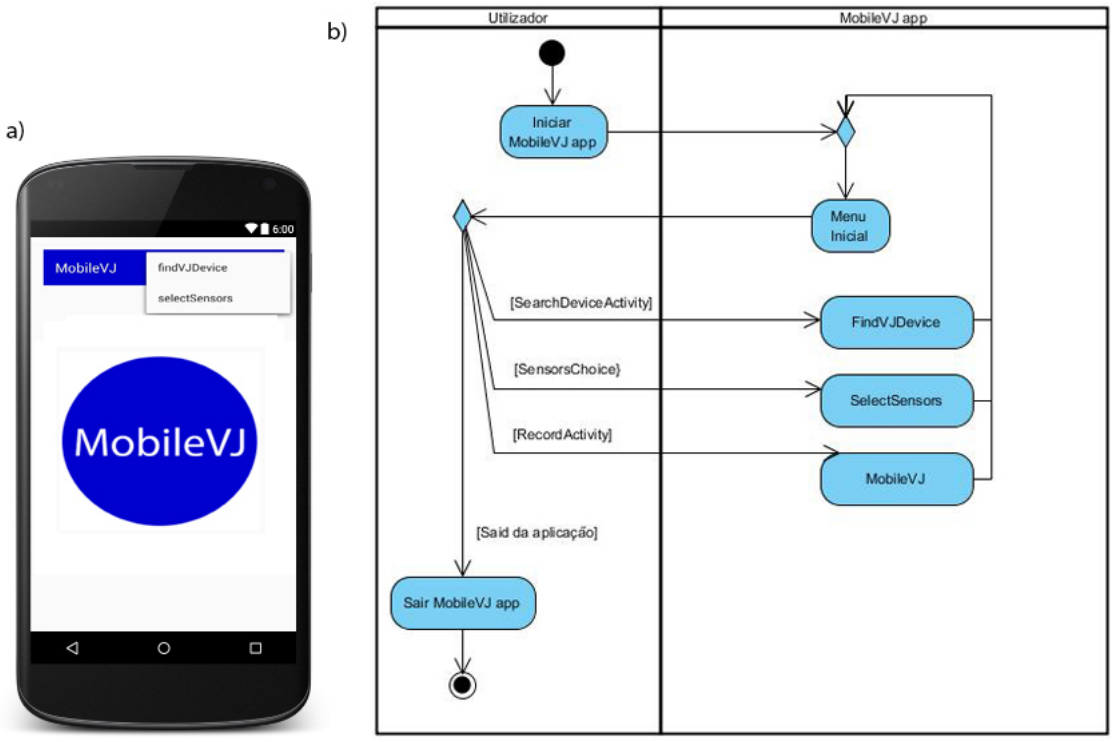

Figura 3.3: a) Print do Menu Inicial da aplicação MobileVJ b) Fluxo de atividade do Menu Inicial da aplicação MobileVJ.

Ao premir a opção “FindVJDevice” do Menu Inicial dá-se o início de uma nova atividade (Figura 3.4 a) cujo objetivo é o utilizador selecionar um dispositivo VJ que se encontra emparelhado com o nosso dispositívo móvel. Esta Atividade (Figura 3.4 b) oferece uma lista com os dispositivos emparelhados por Bluetooth com o dispositivo a utilizar e o utilizador seleciona o *VJ* a que pretende conectar. Ao pressionar no botão "OK", o *MAC Address* do dispositivo selecionado será enviado para o Menu Principal onde será verificada a ligação com o mesmo.

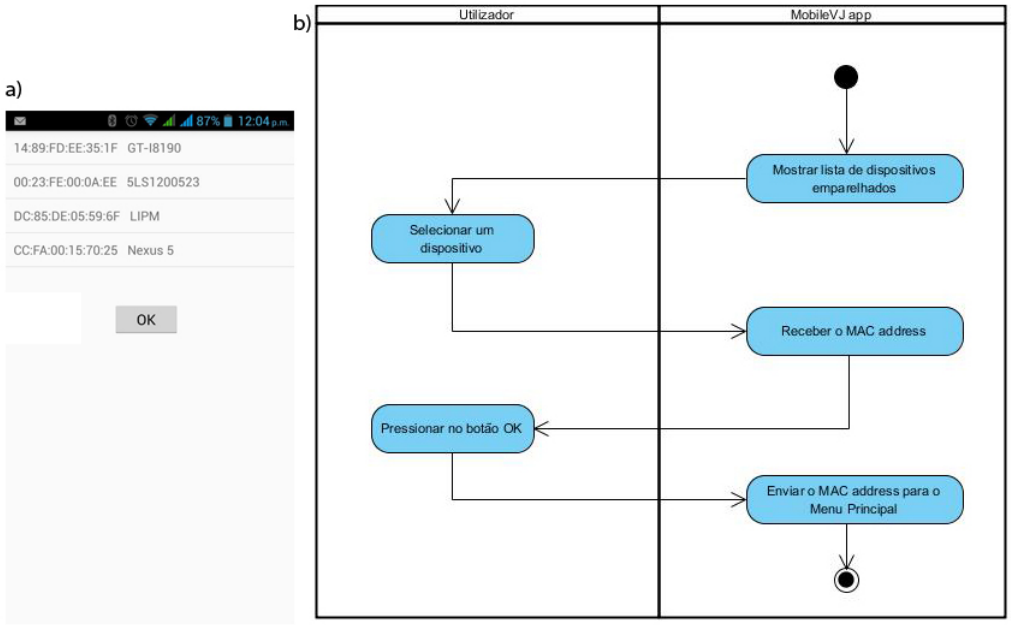

Figura 3.4: a) Print da atividade de seleção do VJ, b) Fluxo de atividades para a seleção do VJ.

Ao premir a opção “selectSensors” do Menu Inicial, dá-se o início a uma nova atividade da seleção dos sensores do próprio smartphone (Figura 3.5 a), Esta atividade oferece uma lista de sensores existentes que o SO Android suporta e um botão de confirmação de sensores selecionados. É permitido a seleção de múltiplos sensores ao mesmo tempo que informação será enviada para os profissionais de saúde, ao pressionar nos elementos de lista e depois pressionando no botão de confirmação, com os nomes dos sensores serão adicionados à lista, e por fim a mesma será enviada para o menu principal (Figura 3.5 b).

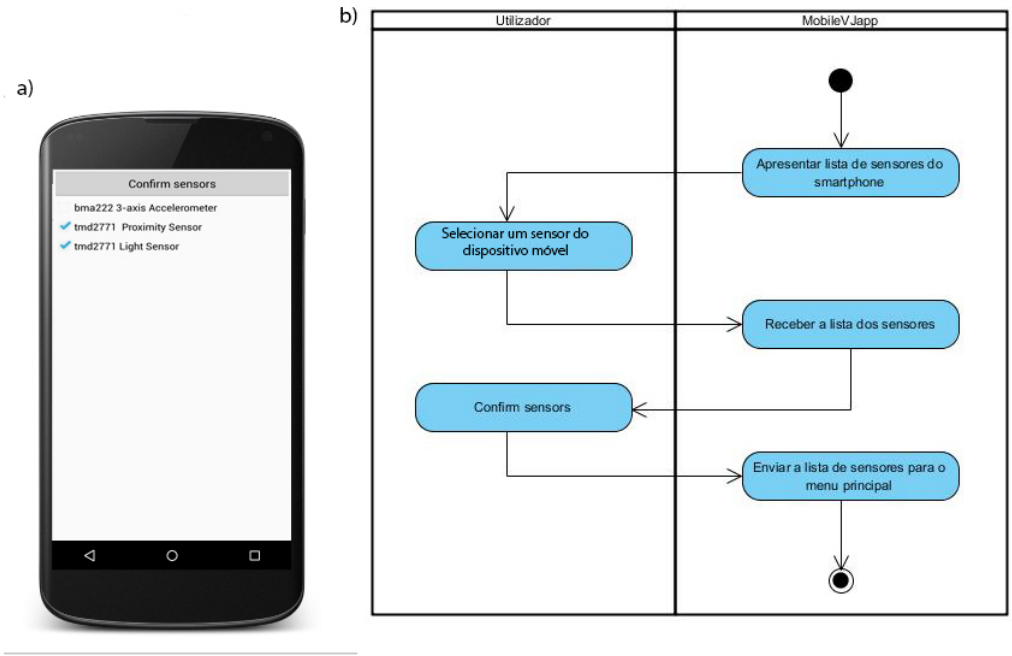

Figura 3.5: a) Print da Atividade de Seleção de sensores do dispositivo Android b) Fluxo de atividades para a seleção de sensores do dispositivo Android.

Ao pressionar no botão *"Mobile VJ"* é verificado se existe uma conexão *INTERNET* no dispositivo, pois é um fator obrigatório nesta dissertação, como também a lista dos sensores do dispositivo móvel ou o MAC address do dispositivo a conectar não se encontrar vazia. Depois destas verificações será iniciada uma nova atividade, onde será verificada a ligação com o servidor *RabbitMQ*, o segundo passo será verificar a lista dos sensores a utilizar e preparar os serviços dos sensores, o terceiro passo será verificar se o MAC address do dispositivo VJ não se encontra vazio, caso exista o MAC address é verificada a conexão com o dispositivo externo. Por fim, a informação dos sensores e do dispositivo externo será exposta nas lables dinâmicas para demonstrar os dados que são recebidos dos sensores e dos dispositivos móveis, depois esta é enviada no formato JSON para o servidor RabbitMQ onde a mesma será distribuída pelos subscribers. Caso a monitorização chegue ao fim do tempo desejado, o utilizador pode carregar no botão *stop*, onde todos os serviços serão terminados. O fluxo desta atividade está demonstrado na Figura 3.6a)

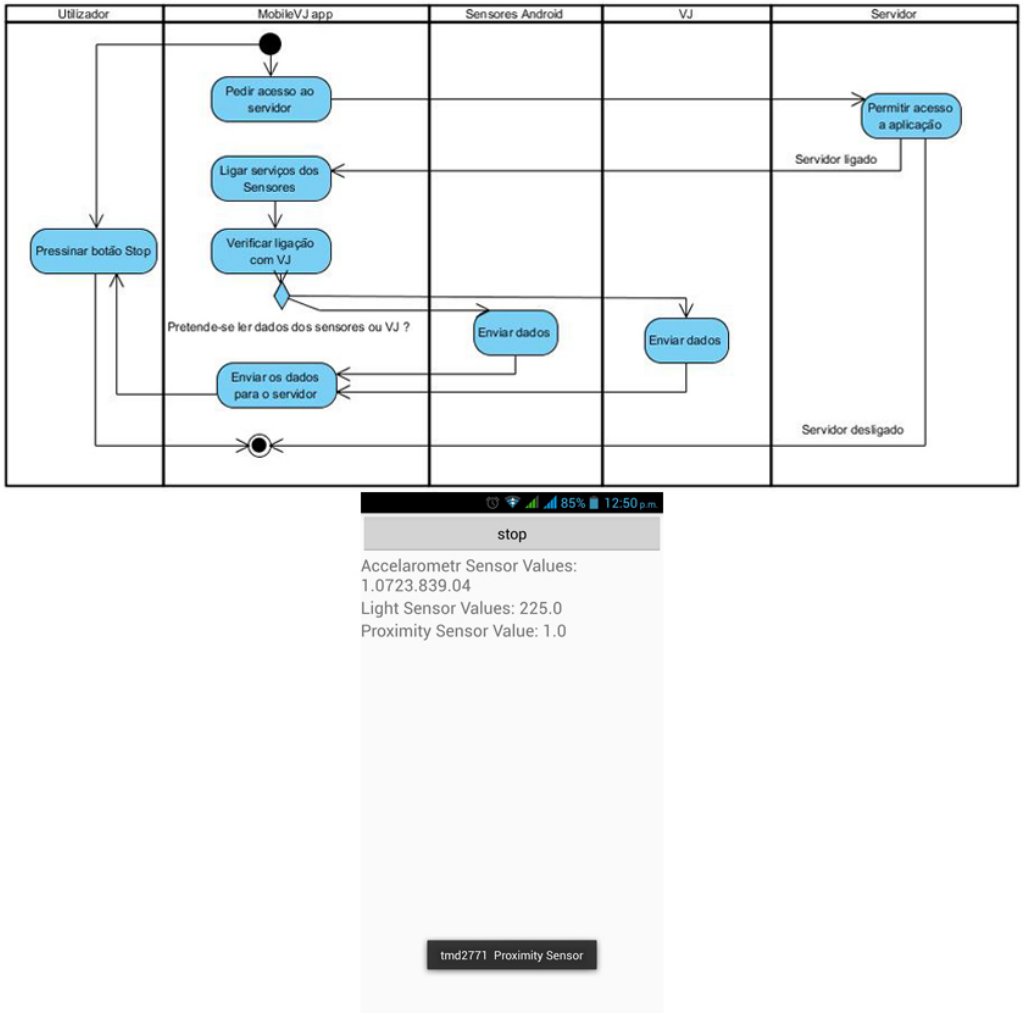

Figura 3.6: a) Print da Atividade de Seleção de sensores do dispositivo Android b) Fluxo de atividades para a seleção de sensores do dispositivo Android.

Para testar a monitorização em real time será necessário criar a aplicação subscriber para testar API criada e projetada para que ao ser acedida, apareça um Menu Inicial (Figura 3.7 a) com uma lista de *RoutingKeys* que a aplicação MobileVJ suporta de momento, uma *checkbox* que permite fazer a escolha entre a aplicação cliente que vai correr no servidor ou não. A aplicação que se pretende correr no servidor vai guardar todas as mensagens recebidas numa base de dados noSQL, enquanto que o cliente genérico apenas receberá mensagens que lhe interessam em *real time* e analisá-las em real time. Existe o botão "Start Service"que apenas será ativado quando pelo menos uma das *RoutingKeys* for escolhida. Este menu corresponde as user stories US001, US201, US202 e US003.

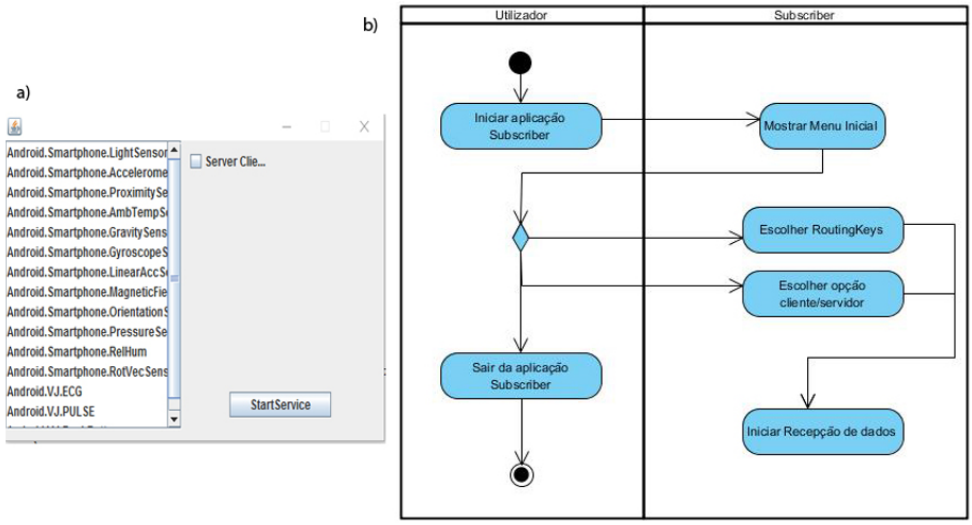

Figura 3.7: a) Print da Atividade de Seleção de sensores do dispositivo Android b) Fluxo de atividades para a seleção de sensores do dispositivo Android.

Após ter pressionado no botão "Start Service"do menu principal, aparece uma janela vazia. Quando as mensagens com as RoutingKeys selecionadas são enviadas pelo emissor publisher, o cliente subscriber receberá estas mensagens, e as mesmas serão visualizadas na janela e guardadas no log.txt.

Para realizar as user stories US101 à US106, foi utilizado o plugin *rabbitmq-management-3.6.2* que foi instalado no computador onde está a correr o servidor rabbitMQ. Este plugin permite realizar todas as User Stories definidas anteriormente.

Para aceder a página de configurações é preciso abrir a página *localhost:15672* (Figura 3.8) onde tem de usar as credenciais utilizador por defeito que será o *guest* e a palavra-passe também será *guest*.

Ao aceder a área ADMIN, serão demonstradas duas áreas: área de *All users* e a área *Add a user*.

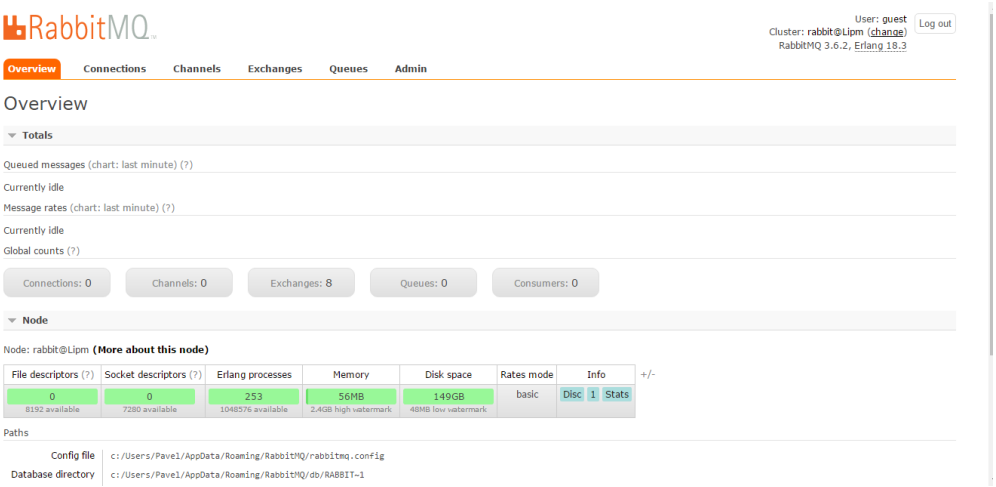

Figura 3.8: Página de gestão do servidor rabbitMQ

Na área de *All users* (Figura 3.9 a) serão demonstrados todos os utilizadores que existem no sistema MobileVJ: o administrador, o cliente e o publisher. Para alterar as permissões de utilizadores criados ou removê-los, basta premir num deles e escolher a opção desejada que aparece numa nova página (figura 3.8 b).

A opção add user permite criar mais utilizadores caso exista uma necessidade de diferenciação.

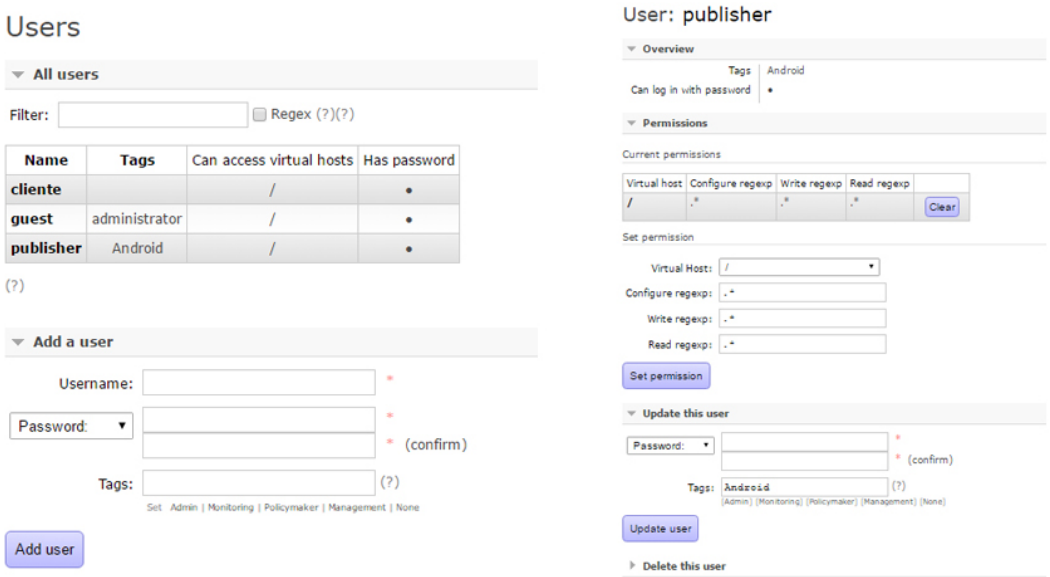

Figura 3.9: a) Página de gestão de todos os utilizadores b) Gestão de um utilizador específico

Ao aceder à área Queues, será demonstrada a lista de Queues criadas pelos clientes. Como administrador é possível remover as Queues dos clientes. Ao aceder à área Exchanges será demonstrada a lista de Exchanges criadas por defeito pelo servidor rabbitMQ e exchange criada pelos dois tipos de clientes ao correr as aplicações deles, será a Exchanges SERVERINESC, que também pode ser removida pelo administrador.

3.5 Arquitetura do sistema

O desenvolvimento de um sistema completo como MobileVJ com a possibilidade de monitorização que seja capaz de agregar informação oriunda dos sensores existentes num dispositivo móvel com a informação recolhida dos dispositivos de monitorização ligados por *bluetooth* como o VJ por exemplo. Este sistema terá um grande impacto na área de saúde, ou profissões de perigo, pois permite a monitorização dos utentes em tempo real.

A aplicação móvel deve conter um módulo que será capaz de agregar a informação recolhida pela aplicação não só pelos sensores incluídos nos dispositivos Android, mas também pelos dispositivos externos ligados a ela por bluetooth. De modo a que ocorra a comunicação entre o dispositivo Android e o dispositivo externo, foram utilizados os SDK externos, capazes de ler as mensagens que serão recebidas pelos dispositivos. Para este projeto foi utilizado o SDK do dispositivo VJ. Este módulo tem em conta que todos os dispositivos Android possuam um diferente número de sensores incorporados, e para que não sejam invocados os sensores inexistentes e os sensores que não são suportados pelo sistema operativo Android.

Uma grande vantagem desta aplicação é a capacidade de agregar os dados de diferentes fontes de informação em real time sem perda de dados e ainda a capacidade de criar uma relação temporal entre os mesmos, quando a aplicação deteta o registo de dados de um sensor. O sistema operativo Android permite a leitura de dados de todos os sensores em simultâneo com o tipo definido `TYPE_ALL` ou então de um sensor de cada vez (ex: `TYPE_ACCELEROMETER`). Aos utilizadores do sistema MobileVJ pode não interessar a monitorização de todos os sinais não só por desnecessidade, mas também como uma forma de poupar a energia do dispositivo. Para isso, será necessário desenvolver os serviços que implementam as funcionalidades de SensorEventListener, que serão capazes de registar toda a informação que o sensor lhe forneça. Estes dados serão enviados para a classe SensorResultReceiver que funciona como o canal de comunicação entre a atividade principal e o SensorService, onde os dados serão analisados e agrupados e depois serão enviados para a classe SensorReceiver da atividade RecordActivity.

Os dados recebidos pelos dispositivos externos serão analisados no código oferecido pelo SDK deles. No caso de VJ, o dispositivo envia diferentes tipos de mensagens que são organizados por cada tipo de sensor que o VJ possui.

Depois de ter registado a alteração de dados, a aplicação MobileVJ cria um JSON e o pacote de dados será enviado para o servidor RabbitMQ (Figura 3.11). O publisher só pode enviar mensagens para um exchange. Um exchange de um lado, recebe mensagens do publisher e do

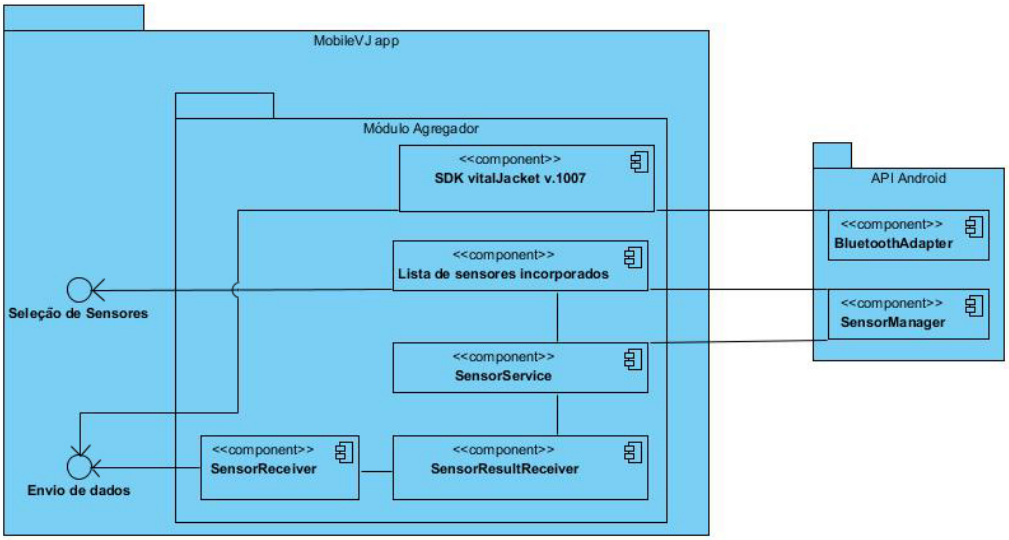

Figura 3.10: Arquitetura do módulo agregador de sensores no sistema MobileVJ

outro reenvia-as para as queues que serão criadas pelos subscribers. No sistema MobileVJ foi utilizado o tipo "topic"onde o exchange reenvia as mensagens para as queues, caso a Routing Key do subscriber corresponda ao Routing Key da mensagem enviada pelo publisher. Para que exista um controlo de toda a informação, ficou decidido criar um cliente do sistema MobileVJ que corra no servidor. A função deste será a recepção de todos os pacotes de dados que serão guardados numa base de dados noSQL.

Caso existam subscribers interessados na informação partilhada pelos publishers, serão criadas as queues de espera pelos mesmos, que funcionam como a ligação entre o subscriber e o servidor. A informação que chega ao cliente final é recebida e guardada num ficheiro ".txt"e é demonstrada numa janela da aplicação subscriber para provar que existe a comunicação entre o publisher e o subscriber. Deste modo este projeto pode extender-se para a monitorização de diferentes tipos de informação, desde a temperatura ambiental recebida pelos dispositivos Android, até o ECG que é recebido por dispositivos externos, como o VJ.

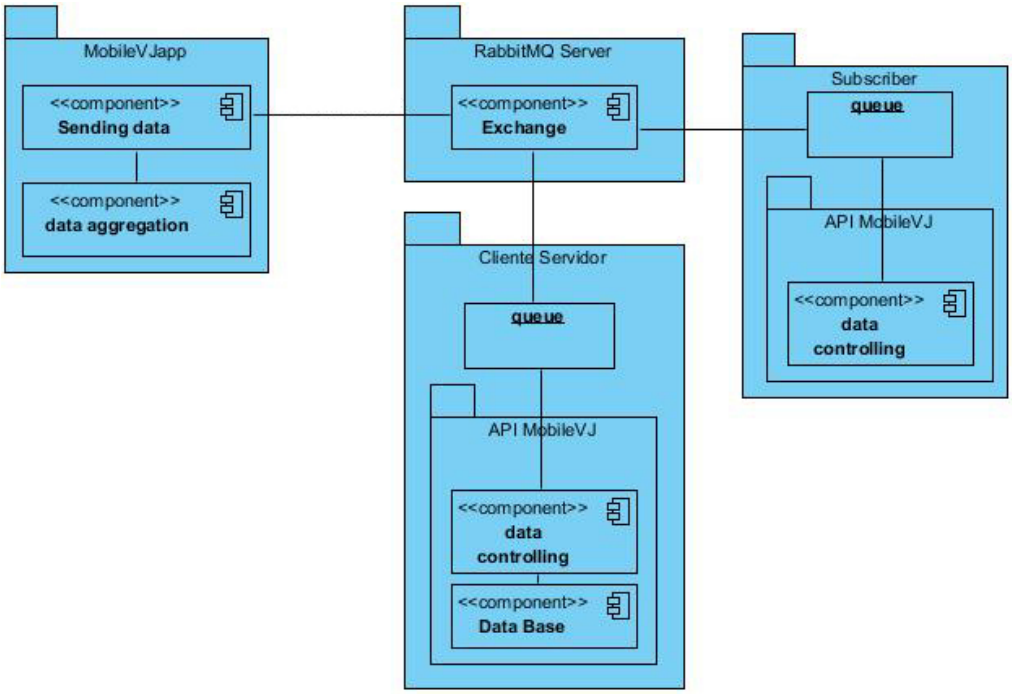

Figura 3.11: Arquitetura da comunicação do sistema MobileVJ entre o publisher e subscriber

3.6 Ambiente de teste

Durante e após o desenvolvimento do sistema MobileVJ pretende-se verificar a sua funcionalidade e usabilidade. Uma vez que o sistema MobileVJ é um sistema complexo, deve-se de ter em conta o material que será utilizado nesta etapa assim como os seus objetivos.

Como o sistema MobileVJ tem três tipos de atores (utilizador monitorizado, profissional de saúde e administrador do servidor) em que cada um deles efetua a sua funcionalidade num dispositivo diferente, será necessário preparar o material de trabalho de cada um deles. O diagrama de Set Up deste sistema pode ser visualizado na Figura 3.12.

Como o utilizador monitorizado será o sujeito em monitorização, este terá de utilizar um dispostivo Android com a aplicação MobileVJ. Deverá possuir acesso à internet, à tecnologia de bluetooth, e ainda possuir mais de que um sensor incorporado. O dispositivo Android a ser utilizado será o smartphone BQ Aquaris 5 HD com um processador MediaTek Quad Core Cortex A7 a 1,2 GHz e com três sensores incorporados: bma222 3x-axis Accelerometer, tmd2771 Proximity Sensor, tmd2771 Light Sensor.

Neste ponto pretende-se testar o sistema MobileVJ. O dispositivo terá acesso à rede sem fios (Wi-fi) e à rede de dados móveis. Os dados de rede móvel a utilizar serão: EDGE (Enhanced Data Rates for GSM Evolution) e Evolved HSPA (High Speed Packet Access).

A tecnologia EDGE foi desenvolvida em 2004 e foi criada para efetuar a transmissão de grande quantidade de dados com uma velocidade máxima de 472 kbps. Contudo os utilizadores apenas conseguem experenciar as velocidades médias entre os 80 kbps e os 130 kbps.

A tecnologia HSPA Evolved aumenta as taxas de transmissão de dados do HSPA fornecendo 42 Mbit/s de download e 11 Mbit/s de upload no uplink cujas taxas de transmissão são até 14,4 Mbit/s para download e 5,76 Mbit/s para upload.

O utilizador subsciber também terá de utilizar um dispositivo externo capaz de ligar o smartphone por bluetooth. Este será o VJ. Este dispositivo é capaz de processar os dados de ECG que recebe e extrair a informação da pulsação e as características de batimentos cardíacos individuais para auxiliar um detetor simples de arritmias.

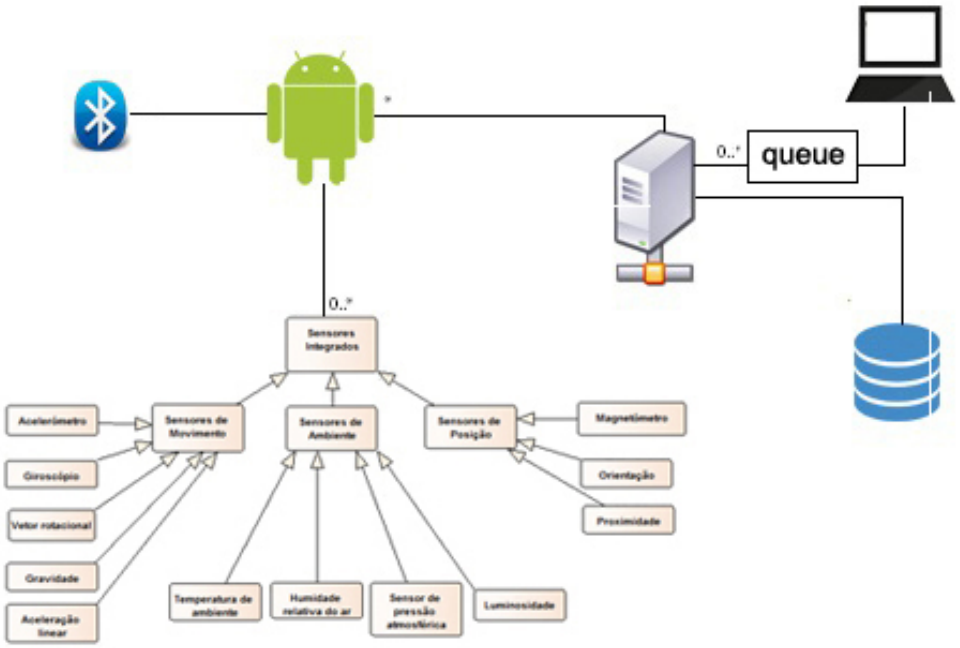

Figura 3.12: Diagrama de Set Up

Como o professional de saúde será o elemento a monitorizar, este precisa de utilizar um computador fixo ou um computador portátil, pois trata-se de um elemento que irá recolher a informação dos dados enviados pelo utilizador monitorizado. Deste modo, este precisa de um aparelho com maior capacidade de processamento de dados em vez de um simples smartphone. O portátil a utilizar foi o Asus k55VM com o processador Intel® Core™ i7-3610QM Quad Core e memória operativa de 6Gb.

Uma vez que o administrador do sevrvidor precisa de ter uma máquina onde deverá ser instalado o servidor rabbitMQ, foi decidido a utilização de um computador pessoal, pois pode-se testar a transmissão de dados entre as diferentes redes. Neste aparelho também deverá de ser instalado a base de dados noSQL mongoDB. Neste computador correrá o software de cliente subscriber que vai ler todas as mensagens e guardá-las na base de dados mongoDB.

3.7 Detalhes da Implementação

Neste ponto serão abordados os elementos de implementação não só da aplicação móvel para o utilizador monitorizado e da aplicação para os profissionais de saúde, mas também será apresentada a configuração do servidor rabbitMQ para que possa ser acedido pelas redes externas.

3.7.1 Implementação de aplicação móvel para utilizador monitorizado

Para o desenvolvimento da aplicação de demonstração da aplicação móvel foi utilizado o ambiente de desenvolvimento Android Studio 2.0. Na criação do projeto na plataforma Android foi determinado que a aplicação a desenvolver correria em dispositivos com API mínima de nível 17, o que corresponde a dispositivos com a versão Android 4.2, mas o alvo seria smartphones Android com API de nível 23, contendo a versão 6.0 deste sistema operativo.

Como o sistema MobileVJ será um sistema de monitorização, de forma a possibilitar a incorporação das funcionalidades pretendidas, a aplicação terá de ter as seguintes permissões adicionadas ao seu ficheiro manifest como é demonstrado na Figura 3.13.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.INTERNET" />
```

Figura 3.13: Permissões introduzidas no ficheiro AndroidManifest.xml da aplicação móvel.

A primeira permissão "ACCESS_NETWORK_STATE"permite à aplicação saber o estado corrente de conexão à internet. Esta opção é de grande importância num sistema de monitorização em tempo real, caso não exista a conexão à internet, não pode existir a transferência de dados em tempo real.

A permissão “BLUETOOTH_ADMIN” possibilita a pesquisa e emparelhamento com aparelhos externos caso um dos dispositivos externos não se encontra emparelhado previamente.

A permissão "BLUETOOTH"possibilita obter a permissão para estabelecer uma conexão com um aparelho externo já emparelhado, aceitar uma conexão e transferir dados. Esta permissão será utilizada para obter a conexão com os dispositivos externos e receber os dados recolhidos por eles.

A última permissão será “INTERNET” e permite que a aplicação MobileVJ utilize sockets. Esta terá um papel importante para estabelecimento de comunicação entre o dispositivo Android e o Servidor e o envio das próximas mensagens com os dados recolhidos.

Para esta aplicação foram adicionadas duas bibliotecas externas (Figura 3.14). A primeira biblioteca "biolib.sdk.jar"é o SDK disponibilizado pelo site oficial do VJ, que oferece as funcionalidades já implementadas como conexão ao VJ, saber o estado de conexão e receber os dados lidos pelo dispositivo. A segunda é "rabbitmq-client.jar"disponibilizado pelo site oficial de rabbitMQ. Esta biblioteca será utilizada para permitir a conexão com o servidor rabbitMQ e o envio das mensagens do dispositivo móvel para o servidor.

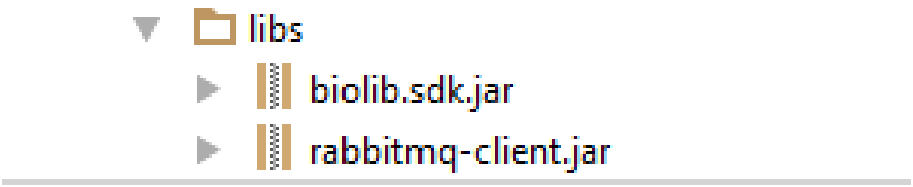

Figura 3.14: Bibliotecas externas que foram adicionadas ao projeto MobileVJ.

A aplicação mobileVJ inclui 4 atividades principais: MainActivity, RecordActivity, SearchDeviceActivity e SensorChoice. Estas têm de ser previamente definidas no ficheiro "AndroidManifest.xml".

A primeira atividade a ser invocada será a MainActivity. Esta será responsável pela intercomunicação de informação entre as outras atividades. Ao ser invocada esta aplicação vai buscar o Secure.ANDROID_ID (Figura 3.15) que é um número de 64 bits (uma string hexadecimal) gerado aleatoriamente quando o dispositivo Android é configurado pela primeira vez. Este deve permanecer constante durante a vida útil do dispositivo e será utilizado como o id único para identificar o publisher no sistema MobileVJ.

```
m_androidId = Settings.Secure.getString(getContentResolver(), Settings.Secure.ANDROID_ID);
```

Figura 3.15: Código que permite buscar o ANDROID_ID único.

Também é utilizado uma parte do handler de SDK de VJ no MainActivity, pois este SDK contém o controlo tanto de bluetooth do dispositivo Android, como o controlo de VJ. Nesta atividade o handler terá a função de verificar o estado de ligação de bluetooth. Para invocar as atividades SearchDeviceActivity e SensorChoice é chamado o método startActivityForResult(Intent Atividade1ToAtividade2, int requestCode) pois estas têm de retornar os resultados que se estão à espera na atividade principal.

Para que possa ser possível receber a informação retornada pelas atividades invocadas, é utilizado o método onActivityResult(int requestCode, int resultCode, Intent data) (Figura 3.14) que fica à espera do requestCode que foi enviado pelo metodo startActivityForResult() e a váriavel resultCode que serve para poder diferenciar tipos de resultado recebidos. Neste projeto o requestCode foi definido de 2 tipos, o primeiro tipo é BioLib.REQUEST_ENABLE_BT e o 0. O primeiro retorna resultCode que verifica a ligação de bluetooth (ex (resultCode == Activity.RESULT_OK). O segundo diz respeito ao retorno de informação das atividades criadas na aplicação móvel. O resultCode esperado é de dois tipos: CHANGE_MACADDRESS e SENSORS_SELECTED. O primeiro demonstra que o resultado recebido foi da atividade SearchDeviceActivity e o resultado retornado será uma string que é um MAC address do VJ. Este valor de retorno é guardado na variável local "address"de tipo String. A segunda, será o resultado recebido da atividade SensorsChoice

que retorna um array de strings com os nomes dos sensores que serão guardados na variável local mUsesList de tipo *Vector<String>*. Para que possa ser possível guardar a variável de retorno, foi realizado um cast de tipo de variável de ArrayList para Vector de tipo String (Figura 3.17).

```
Intent myIntent = new Intent(MainActivity.this, SearchDeviceActivity.class);
startActivityForResult(myIntent, 0);
```

Figura 3.16: Segmento de código que será utilizado para invocar uma nova atividade.

```
mUsesList.addAll(Arrays.asList(data.getExtras().getStringArray(SensorsChoice.SELECT_SENSORS)));
```

Figura 3.17: Segmento de código que permite fazer o cast do resultado recebido da atividade SensorsChoice.

Para invocar a atividade de transmissão de dados a RecordActivity, em primeiro lugar é chamado o método isOnline() que verifica a conexão à internet do dispositivo Android que está a ser utilizado (Figura 3.18). Depois da aplicação estar conectada à internet, serão verificados os dados necessários para iniciar a RecordActivity com o método criado isRecordActivityReady(). Só após estas duas validações será invocada a atividade RecordActivity e enviada para ela a informação adicional como a lista com os nomes dos sensores a utilizar, o id único do dispositivo Android e o MAC address do VJ.

```
public boolean isOnline() {
    ConnectivityManager cm =
            (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo netInfo = cm.getActiveNetworkInfo();
    return netInfo != null && netInfo.isConnectedOrConnecting();
}
```

Figura 3.18: Código para o sistema operativo Android que permite verificar a conectividade à rede Internet .

A SearchDeviceActivity é a segunda atividade a ser invocada e é responsável pela visualização de todos os dispositivos uma vez já emparelhados com o nosso dispositivo Android e a seleção de um deles. Para implementar estas funcionalidades foi utilizada uma variável de tipo BluetoothAdapter capaz de verificar o estado atual do dispositivo bluetooth, tanto como retornar à lista com os dispositivos previamente emparelhados. Para a visualização dos resultados

foi utilizada uma ListView. Esta implementa o método *setOnItemClickListener(new AdapterView.OnItemClickListener())* que permite selecionar uma das opções da lista dos dispositivos de forma a retornar o MAC address do dispositivo como é demonstrado na Figura 3.19.

```
mainListView.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick( AdapterView<?> parent, View item, int position, long id) {
        selectedValue = (String) listAdapter.getItem(position);
        Intent intent = new Intent();
        intent.putExtra(SELECT_DEVICE_ADDRESS, selectedValue);
        setResult(CHANGE_MACADDRESS, intent);
        finish();
    }
});
```

Figura 3.19: Seleção de um MAC address e o retorno do mesmo para a atividade principal.

A terceira atividade é a SensorsChoice. Esta atividade tem como o objetivo a implementação do código capaz de retornar a lista de sensores presentes no dispositivo que está a ser utilizado e oficialmente suportados pelo sistema operativo Android. O segundo objetivo será retornar os sensores escolhidos para a atividade principal da aplicação MobileVJ.

Para o primeiro objetivo foi utilizada a variável de tipo *SensorManager* capaz de retornar uma lista com todos os sensores presentes no dispositivo. O próximo passo será a verificação de todos os elementos de lista pela função androidSuportedSensors(List<Sensor> list). Esta tem como o objetivo excluir os sensores presentes no dispositivo Android que não estão definidos no *Sensor.TYPE_SENSOR*. Para cumprir o segundo objetivo desta atividade é utilizado um array de CheckBox do mesmo tamanho da lista com os sensores presentes no dispositivo. Ao pressionar no botão de confirmação, cada elemento do array de CheckBox é verificado se é pressionado e os elementos selecionados são adicionados a uma lista com os nomes dos sensores que serão usados na recolha de dados.

Por fim, temos a atividade RecordActivity, que é a aplicação mais complexa. Esta contém vários objetivos do projeto MobileVJ. Os objetivos principais desta atividade serão a agregação de informação recolhida não só pelos sensores mas também por dispositivos externos ligados por bluetooth para envia-los em tempo real para o servidor rabbitMQ.

O início desta atividade dá-se na recolha de dados como MAC address, androidID e a lista com os nomes de sensores por utilizar enviados pela atividade principal MainActivity.

Após ter recebido os dados, a aplicação móvel faz a conexão com o servidor rabbitMQ atráves dos funcionalidades implementadas na livraria "rabbitmq-client.jar". A primeira coisa a fazer será a autenticação com as credenciais definidas pelo administrador criadas no servidor rabbitMQ. O próximo passo será a abertura de canal de comunicação. Para isso é definida a exchange e o seu tipo a utilizar (Figura 3.20).

```
Connection connection = factory.newConnection();
channel = connection.createChannel();
channel.exchangeDeclare(Constants.EXANGE_NAME, Constants.RABBITMQ_TOPIC);
```

Figura 3.20: Ligação ao servidor RabbitMQ.

Após se ter estabelecido a comunicação com o servidor, serão inicializadas as listas de sensores através da lista de nome dos sensores recebidos pela MainActivity. Pois este processo é obrigatório, e como não se pode enviar os dados de tipo Sensor de uma atividade para uma outra, foi escolhida esta alternativa. Através do método implementado *stringToSensors(Vector<String> selectedS, List<Sensor> returnedSensors)* é retornada uma lista de Sensores que serão utilizados na monitorização do sujeito. A seguir é chamado o método *startReceivers()*, cujo objetivo é criar um canal de comunicação entre a atividade RecordActivity e cada um dos serviços criados para recolha de dados dos sensores (Figura 3.21).

```
accelerometerResultReceiver = new AccelerometerResultReceiver(sensorHandler);
accelerometerResultReceiver.setReceiver(new AccelerometerReceiver());
```

Figura 3.21: Exemplo de inicialização de canal de comunicação para o sensor de tipo ACCELEROMETER.

O próximo passo será a criação dos serviços para cada um dos sensores a utilizar. Previamente, todos os serviços têm de ser definidos no ficheiro "AndroidManifest.xml". Os serviços serão inicializados na função startServices(). Para cada um dos serviços de tipo SensorService será enviado um extra de tipo SensorResultReceiver (Figura 3.22).

```
Intent accIntent = new Intent(RecordActivity.this, AccelerometerService.class);
accIntent.putExtra(Constants.REC, accelerometerResultReceiver);
startService(accIntent);
```

Figura 3.22: Exemplo de inicialização de um serviço para recolha de dados do sensor de tipo ACCELEROMETER.

Após a inicialização dos serviços para cada um dos sensores a monitorizar, é inicializada a variável lib_vj de tipo BioLib. Anteriormente, é verificado se o MAC address recebido pela atividade de menu principal não se encontra vazio. Caso o MAC address se encontre vazio, é sinal de que nenhum VJ foi selecionado pelo utilizador e não é preciso afetuar nenhuma recolha de dados do VJ. Caso o MAC address não se encontre vazio, será inicializada a variável lib_vj. O código utilizado para inicializar esta variável será *lib_vj = new BioLib(this, vjHandler);* onde as variáveis a passar como argumentos serão o contexto da actividade RecordActivity e o handler fornecido pelo SDK de VJ. Após ter inicializado a variável lib_vj é inicializada a conexão e a receção de dados recolhidos pelo VJ.

O handler que é disponibilizado pelo SDK do VJ é ligeiramente alterado para que seja possível enviar as mensagens recebidas para o servidor rabbitMQ. Este é separado por vários tipos de mensagens, como MESSAGE_DATA_UPDATED ou MESSAGE_ECG_STREAMAs por exemplo. As mensagens recebidas serão guardadas num JSON com um timestamp associado ao tempo em que a mensagem foi emitida e é enviada para o servidor rabbitMQ através da função *rabbitMQ_publisher(String routingKey, JsonObject json)*.

Na atividade *RecordActivity* têm de ser definidas as classes SensorReceiver que implementarão a interface SensorResultReceiver.Receiver. Estas classes não têm um número de variáveis fixo, dependendo de um sensor para o outro. Tudo depende do tipo de sensor e quantos dados serão retornados pelo sensor. Por exemplo, o acelerômetro mede a força de aceleração em m/s^2 em que estão a ser utilizados todos os eixos físicos x, y, e z, então na class AccelerometerReceiver também serão definidas três variáveis para receber os dados do sensor. A class SensorReceiver tem

uma função newEvent(), que está definida pela class SensorResultReceiver. Esta retorna os dados recolhidos pelos sensores para a atividade RecordActivity. Após os dados serem recolhidos é criado um objeto JSONObject [1] com os dados recolhidos pelo sensor com a informação adicional, como o androidId e o tempo em milésimos de segundo em que o evento foi realizado. Após ter criado o JSON, este é enviado para a função *rabbitMQ_publisher(String routingKey, JsonObject json)*, a partir de onde a mensagem será enviada para o servidor rabbitMQ.

Durante a realização de monitorização é pretendido realizar o envio de dados em tempo real para o servidor. Desta forma foi implementada a função *rabbitMQ_publisher(String routingKey, JSONObject json)* que fica à espera de duas variáveis, a routingkey de tipo String e JSON de tipo JSONObject. Como o rabbitMQ utiliza a transferência de mensagens em Bytes, O objeto JSONObject retorna o JSON que será previamente transformado para a String e só depois disto será transformado em Bytes. A routingkey que a função rabbitMQ_publisher() recebe são enviadas pelas classes SensorReceiver e os handlers dos dispositivos externos. As routing key são as Strings de encaminhamento de mensagens, como por exemplo: *"Android.Smartphone.AccelerometerSensor"*. Esta Routing Key permite ao profissional de saúde ter uma forma mais organizada e mais intuitiva de se subscrever a dados de interesse. A mensagem é enviada pelo canal previamente aberto na abertura de atividade RecordActivity(Figura 3.23).

```
channel.basicPublish(Constants.EXANGE_NAME, path, null, json.getJson().toString().getBytes());
```

Figura 3.23: Mensagem enviada pelo canal de comunicação entre a aplicação e o servidor.

Após a monitorização, é importante desligar todos os serviços que foram utilizados para a monitorização e não esquecer de efetuar a desconexão do dispositivo externo. No caso de VJ, caso a conexão com o dispositivo for interrompida incorretamente, é preciso um tempo para que o VJ seja capaz de se conectar de novo ao dispositivo por bluetooth. Por isso, no listner do botão STOP são verificados os serviços que estão ativos e dispositivos externos que estão ligados à aplicação mobileVJ a que se pretende desligar para que não ocorra um erro na próxima conexão.

Para além de ter atividades definidas, a aplicação contém classes de agregação de dados recolhidos pelos sensores, estes são os serviços referidos na parte de RecordActivity. A class SensorService implementa os métodos definidos pela interface SensorEventListener. No total foram implementados 17 serviços para cada um dos sensores. A classe SensorService implementa a função *onSensorChanged()* onde serão registados as variações de dados recolhidos pelos sensores. Na mesma função os dados recolhidos serão enviados para a classe SensorResultReceiver.

A classe SensorResultReceiver estende da classe da ResultReceiver e serve de um canal de comunicação entre a atividade RecordActivity e o SensorService. Nesta classe é implementada uma interface *Receiver* com a função newEvent(float x, float y, float z). A classe SensorReceiver

[1] JSONObject - Uma classe criada que permite uma gestão e organização de dados recolhidos em formato de JSON.

definida na atividade RecordActivity implementa esta interface da classe SensorResultReceiver. Também é implementado o método *onReceiveResult(int resultCode, Bundle resultData)*, sempre que o sensor do serviço subscrito sofrer alterações, esta será notificada e os dados recolhidos serão enviados para a atividade RecordActivity através da class SensorReceiver.

Para a ligação com o servidor rabbitMQ serão precisas as credenciais, nome de exchange definida e as Routing Keys e para não haver inconsistências ficou decidido a criação de uma classe de constantes (Figura 3.24)

```
public class Constants {

    public static final String LOG_TAG = "myLOG" ;
    public static final String REC = "Receiver" ;
    public static final String EXANGE_NAME = "SERVER_INESC" ;
    public static final String RABBITMQ_TOPIC = "topic" ;
    public static final String RMQ_CRED = "publisher" ;
    public static final String RMQ_IP = "188.81.255.41" ;
    public static final String RMQ_LIGHT_PASS = "Android.Smartphone.LightSensor" ;
    public static final String RMQ_ACC_PASS = "Android.Smartphone.AccelerometerSensor" ;
    public static final String RMQ_PROX_PASS = "Android.Smartphone.ProximitySensor" ;
    public static final String RMQ_AMB_TEMP_PASS = "Android.Smartphone.AmbTempSensor" ;
    public static final String RMQ_GRAVITY_PASS = "Android.Smartphone.GravitySensor" ;
    public static final String RMQ_GYROSCOPE_PASS = "Android.Smartphone.GyroscopeSensor" ;
    public static final String RMQ_LINEAR_ACC_PASS = "Android.Smartphone.LinearAccSensor" ;
    public static final String RMQ_MAGNETIC_FIELD_PASS = "Android.Smartphone.MagneticFieldSensor" ;
    public static final String RMQ_ORIENTATION_PASS = "Android.Smartphone.OrientationSensor" ;
    public static final String RMQ_PRESSURE_PASS = "Android.Smartphone.PressureSensor" ;
    public static final String RMQ_REL_HUM_PASS = "Android.Smartphone.RelHum" ;
    public static final String RMQ_ROT_VEC_PASS = "Android.Smartphone.RotVecSensor" ;
    public static final String RMQ_VJ_ECG_PASS = "Android.VJ.ECG" ;
    public static final String RMQ_VJ_PULSE_PASS = "Android.VJ.PULSE" ;
    public static final String RMQ_PUSH_BUTTON_PASS = "Android.VJ.PushButton" ;
}
```

Figura 3.24: Classe de constantes com as variáveis definidas para a comunicação entre o publisher e o servidor rabbitMQ.

Durante a realização de uma monitorização, a aplicação vai enviar os dados de todos os eventos registados por cada sensor assim como o tempo em milésimos de segundo a que se deu o novo evento e a identificação do dispositivo móvel. De forma a que o profissional de saúde possa consultar as medições posteriores é necessário saber qual o utilizador em monitorização e o sensor correspondente de cada medição. Para isto foi desenvolvida a class JSONObject que permite organizar a informação de forma simples de modo a que o profissional de saúde possa lê-la e guarda-la na base de dados noSQL sem alterações. O método *writeDataToJson(String androidID , String sensor, String time, float a, float b , float c, float d)* permite guardar os dados recolhidos pelos sensores num JSON que fica com a estrutura demonstrada na Figura 3.25.

```
{
    "androidId": "7C0A89B1447293C5",
    "sensor": 1,
    "values": {
        "time:": "1464682578264",
        "valX:": "5.363",
        "valY:": "7.048"
        "valZ:": "4.443",
    }
}
```


Figura 3.25: Classe de constantes com as variáveis definidas para a comunicação entre o publisher e o servidor rabbitMQ

3.7.2 Implementação de aplicação para o profissional de saúde

Para o desenvolvimento da aplicação de demonstração para os profissionais de saúde foi utilizado o software Eclipse Mars 2. Esta aplicação tem como o objetivo correr no servidor ou numa outra máquina de forma a garantir que esta seja capaz receber as mensagens apenas do interesse do utilizador em tempo real.

Para esta aplicação foram adicionadas duas bibliotecas externas. A primeira biblioteca é "rabbitmq-client.jar"que já foi mencionada no ponto anterior. A segunda biblioteca é "mongo.jar"que será utilizada para guardar toda a informação recebida pelo servidor numa base de dados.

A primeira classe a ser criada foi a classe de constantes similar à classe utilizada na aplicação móvel do sistema MobileVJ. No entanto esta sofreu ligeiras alterações que são demonstradas na Figura 3.26.

```
public static final String RMQ_CRED = "guest" ;
public static final String RMQ_CRED_CLI = "client" ;
public static final String RMQ_IP_SERVER = "127.0.0.1" ;
public static final String RMQ_IP = "188.81.255.41" ;
public static final int RMQ_MIN_BIND = 1 ;
```

Figura 3.26: Alterações na class de constantes utilizada na aplicação de utilizador subsciber

Existem dois tipos de credenciais e IP, pois o cliente subscriber pode correr no próprio servidor rabbitMQ ou então numa outra máquina. Para a aplicação cliente que corra no servidor são utilizadas as variáveis RMQ_CRED e RMQ_IP_SERVER. O utilizador com o cliente a correr numa outra máquina vai utilizar as variáveis RMQ_CRED_CLI e RMQ_IP. A variável RMQ_IP guarda o ip do servidor para que possa ser acedido de qualquer ponto de mundo enquanto a variável RMQ_IP_SERVER serve para utilização de localhost.

Toda a informação e o acesso serão controlados pela classe RabbitMQCliente. A primeira etapa a ser executada para os dois tipos de aplicação subscriber é a ligação com o servidor rabbitMQ. A aplicação a correr no servidor utiliza as credenciais e o ip para a máquina local, enquanto a aplicação que será utilizada numa máquina diferente utilizará as credenciais de acesso exterior como é demonstrado na Figura 3.27.

```
if(server){
factory.setHost(Constants.RMQ_IP_SERVER);
factory.setUsername(Constants.RMQ_CRED);
factory.setPassword(Constants.RMQ_CRED);
}
else{
    factory.setHost(Constants.RMQ_IP);
    factory.setUsername(Constants.RMQ_CRED_CLI);
    factory.setPassword(Constants.RMQ_CRED_CLI);
}
```

Figura 3.27: Ligação ao servidor rabbitMQ dependendo de tipo de aplicação.

A partir da ligação ao servidor, o procedimento será diferente. Para a aplicação subscriber a correr no servidor, o próximo passo será a ligação à base de dados com o nome que lhe foi atribuído pelo utilizador. A seguir é retornada a *collection* que será utilizada para guardar os dados recebidos, caso a *collection* ainda não exista, ela será criada.

O próximo passo será criar a queue que fica à espera de mensagens. A Routing Key utilizada por ela será o cardinal, esta permite ler todas as mensagens recebidas pelo servidor rabbitMQ, e como o cliente que corre no servidor pretende guardar todas as mensagens recebidas pelo servidor, é utilizada esta Routing Key.

Após a criação da queue é criado o consumer que vai receber todas as mensagens. Quando a mensagem é recebida, esta sofre uma conversão de formato JSON para DBObject que permite a inserção direta da mensagem recebida (Figura 3.28).

```
DBObject dbObject =(DBObject)JSON.parse(message);
coll.insert(dbObject);
```

Figura 3.28: Inserção de mensagem recebida para a base de dados mongoDB

Para a aplicação subscriber, que efetua a monitorização, o próximo passo será criar queues. Uma queue é associada a uma Routing Key e desta forma serão criadas tantas queues quantas Royting Keys foram selecionadas pelo utilizador.

Após as criação das queues é inicializado o consumer que vai receber todas as mensagens recebidas. As mensagens recebidas são visualizadas na janela da aplicação subscriber. Isto seria apenas uma demonstração das mensagens que estão a ser recebidas em tempo real, da mesma

forma que as mensagens poderiam ser vizualizadas num gráfico se isto for pretendido ou enviadas para uma outra aplicação. As mensagens recebidas pelo subscriber podem ser consultadas no Anexo A.

3.7.3 Configuração do servidor

Para atingir os objetivos desta dissertação, foi preciso configurar o servidor para que este possa estabelecer a comunicação entre o cliente publisher e cliente subsciber em real time e guardar todas as mensagens numa base de dados.

Para correr o servidor rabbitMQ, foi preciso previamente instalar o Erlang Windows Binary File que está disponibilizado pelo site "http://www.erlang.org/". Após a instação do Erlang, deve-se correr o executável "rabbitmq-server-3.6.2"disponibilizado no site oficial de rabbitMQ. Depois de ter o servidor rabbitMQ instalado, foi instalado o plugin "rabbitmqadmin". Este permite o controlo de tráfego das mensagens, de queues criadas e de utilizadores (Figura 3.8).

Para que o servidor possa ser acedido de qualquer ponto do mundo, será preciso abrir as portas utilizadas por defeito pelo rabbitMQ 5671 e 5672 responsáveis pela comunicação TCP entre o servidor e dispositivos ligados a ele.

No servidor também será instalada a base de dados noSQL o mongoDB. A versão instalada no servidor foi "MongoDB for Windows 64-bit"disponibilizada pelo site oficial de mongoDB. Em seguida será criado a pasta db, onde se pretende guardar todos os dados recolhidos pelos sensores de todos os monitorizados em questão.

3.8 Conclusão

O sistema MobileVj pretende oferecer um sistema especializado de monitorização de utilizadores em tempo real que se encontram à distância dos professionais de saúde. Para isso, foi necessário a criação de uma aplicação publisher na plataforma Android, que fosse capaz de recolher em simultâneo os dados obtidos pelos dispositivos externos, como o VJ, com os sensores presentes no dispositivo móvel utilizado. Como também, a criação de uma aplicação subscriber capaz de receber as mensagens com interesse para a monitorização.

Numa primeira fase foi necessário definir os requisitos que o sistema MobileVJ devia cumprir, os atores e as user stories que definem o que cada um dos atores pretende realizar, assim como o fluxo de atividades a que deviam obedecer as aplicações.

Numa segunda fase, foi definida a arquitetura da aplicação móvel e de sistema MobileVJ em questão. Esta permitiu escolher os devices que serão necessários para testar o sistema MobileVJ.

Por fim foi configurado o servidor para que possa correr o servidor rabbitMQ de forma a poder ser acedido em qualquer ponto de mundo como também foram criadas as aplicações publisher e subscriber, capazes de realizar a monitorização do sujeito em monitorização.

Capítulo 4

Resultados e Discussão

MobileVJ é um sistema que pretende oferecer a monitorização de utilizadores à distância em tempo real. Para isto foram desenvolvidas as aplicações publisher e subscriber com o intuito de demonstrar e comprovar o funcionamento do sistema criado no âmbito desta dissertação, no que toca à capacidade de recolher e associar os valores registados por sensores tanto do dispositivo móvel como de dispositivos externos e envia-los para os utilizadores subscribers interessados neles.

A aplicação publisher deve ser capaz de conseguir receber os dados recolhidos pelos dispositivos externos e os sensores do smartphone. Posteriormente, esta deve ser capaz de enviar os dados sem atrasos [1] para o servidor rabbitMQ.

A aplicação subscriber deve ser capaz de permitir a leitura de dados de mais de que um sensor enviados pelos diferentes utilizadores de tipo publisher.

Para comprovar o funcionamento do sistema MobileVJ foram realizados 4 testes. Nos primeiro testes na aplicação publisher foram utilizados diferentes tipos de rede cuja velocidade será diferente. A experiência realizada serve para perceber o impacto da velocidade da internet em relação à troca das mensagens entre o cliente publisher e o cliente subscriber. O segundo teste serve para demonstrar o funcionamento de multi-monitorização onde dois clientes de tipo subscriber enviam a informação recolhida para um cliente de tipo subscriber, um tempo depois será adicionado um segundo e terceiro cliente para verificar o comportamento do sistema em relação às alterações e serão também comparadas as *timestamp* de envio e recepção de mensagens para verificar os atrasos possíveis no sistema de monitorização.

Foi utilizada a página de gestão no servidor rabbitMQ, que permite controlar o fluxo de mensagens recebidas.

Os últimos 2 testes realizados servem para comprovar o funcionamento do sistema e capacidade do mesmo em termos de uso de monitorização de sinais vitais de tipo ECG e Pulso. No primeiro teste, o monitorizado estará em estado de repouso (sentado) e durante 15 minutos serão recolhidos os dados vitais através da camisola VJ. No segundo teste os dados são recolhidos durante um exercício de levantamento de pesos para o trícep, de modo a verificar a variação de sinais

[1] O atraso representa o tempo entre o envio e a receção de mensagem

vitais com o exercício físico. O exercício a ser realizado foi de 3 séries com 20 repetições cada uma.

4.1 Teste 1

A velocidade da internet tem uma influência na resposta dos seus serviços. O MobileVJ visa ser um sistema de monitorização em tempo real, e como o sistema depende diretamente da rede de internet, foi então decidido testar a partir de que velocidade de internet seria mais eficaz e o tipo de rede que não criasse atrasos na transferência de dados.

Para este teste foram utilizados os dados recolhidos pelo dispositivo VJ, o acelerómetro, o sensor de luz e o sensor de proximidade incluídos no dispositivo móvel a utilizar. Também foram testados três tipos de rede: EDGE, Evolved HSPA e Wi-Fi.

Antes de começar os testes, foi efetuada a medição da velocidades de internet que cada uma das redes oferece. Para a sua medição foi utilizada a aplicação Speedtest. Os resultados obtidos podem ser comparados na tabela 4.1.

Tabela 4.1: Tabela com comparação de velocidades obtidas pela rede Wi-Fi, Evolved HSPA e EDGE.

Tipo de rede	Velocidade Download	Velocidade Upload
EDGE	0.14 Mbps	0.06 Mbps
Evolved HSPA	5.95 Mbps	3.42 Mbps
Wi-Fi	37.83 Mbps	20.34 Mbps

O protocolo dos testes é demonstrado na Figura 4.1.

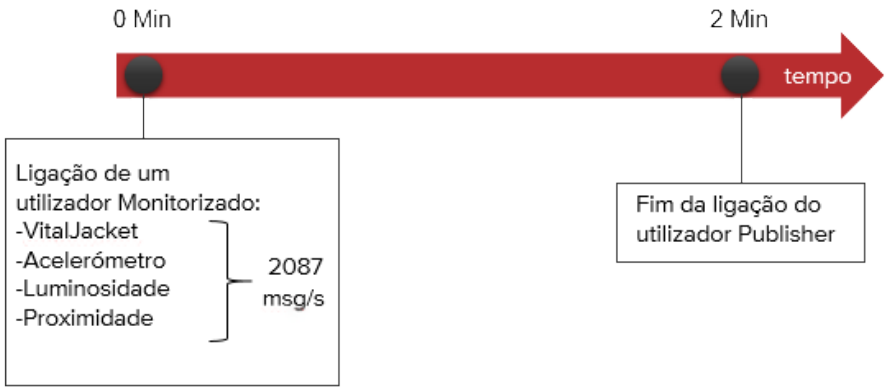

Figura 4.1: Protocolo do Teste Nº1

A primeiro iteração deste teste foi realizada com a rede EDGE ativada no dispositivo móvel para correr a aplicação móvel desenvolvida. Na aplicação de tipo publisher foram ativados todos os sensores presentes no smartphone bq 5hd e lidos os dados do dispositivo VJ. Os resultados obtidos são demonstrados nas Figuras 4.2 e 4.3

Figura 4.2: Velocidade de recepção de mensagens no momento na rede EDGE

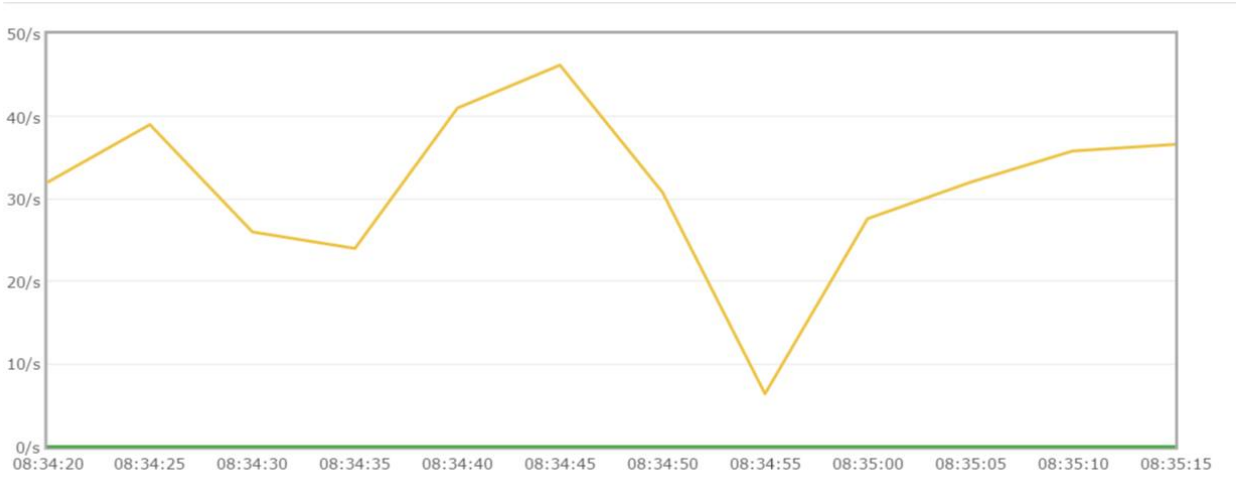

Figura 4.3: A variação de fluxo de mensagens durante um minuto a utilizar a rede EDGE

Como é demonstrado na figura 4.3, ao utilizar a rede EDGE o fluxo das mensagens não é constante, isto quer dizer que alguns dos conjuntos de dados da aplicação móvel com este tipo de rede demoraria a enviar. Portanto, a rede EDGE é inapropriada no uso da aplicação móvel desenvolvida.

A segunda iteração do teste já foi realizada com uma rede de tecnologia 3g a Evolved HSPA. As restantes condições do teste mantiveram-se constantes. Os resultados deste teste podem ser visualizados pela figura 4.4 e 4.5.

Figura 4.4: Velocidade de recepção de mensagens no momento na rede Evolved HSPA.

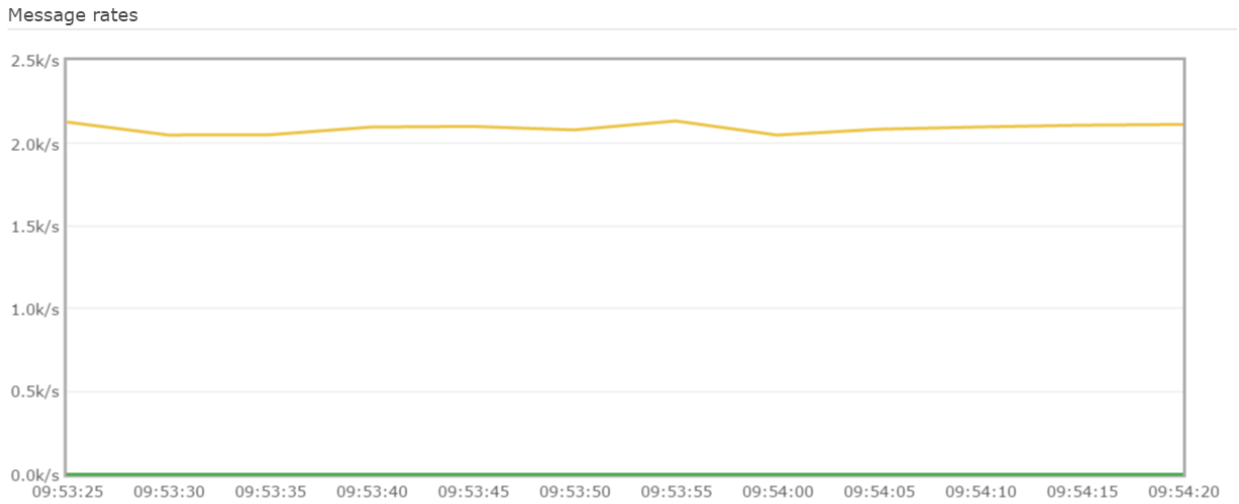

Figura 4.5: A variação de fluxo de mensagens durante um minuto a utilizar a rede Evolved HSPA.

Como é demonstrado na figura 4.5, ao utilizar a rede Evolved HSPA o fluxo das mensagens é constante, apesar de que ao fim de um minuto ocorreu uma oscilação de fluxo das mensagens, devido ao acelerómetro e sensores de proximidade que enviam as mensagens quando existe apenas uma alteração nas suas medições. Isto quer dizer que a rede Evolved HSPA é apropriada para ser utilizada pela aplicação móvel.

A terceira iteração do teste foi realizada com uma rede Wi-Fi. As condições do teste mantiveram-se constantes. Os resultados deste teste podem ser visualizados nas Figuras 4.6 e 4.7.

Figura 4.6: Velocidade de recepção de mensagens no momento na rede Wi-Fi.

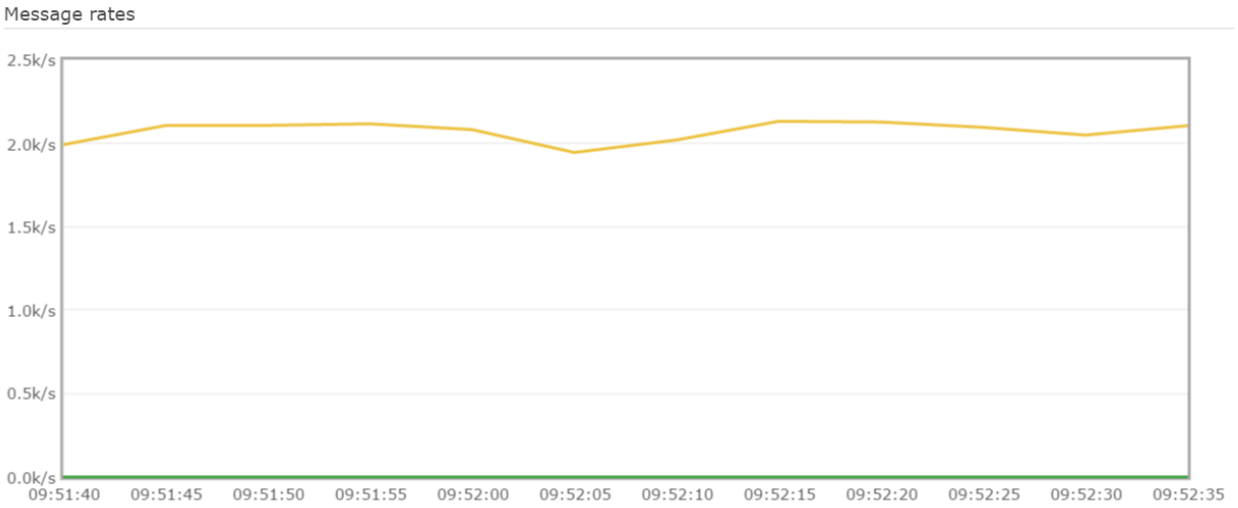

Figura 4.7: A variação de fluxo de mensagens durante um minuto a utilizar a rede Wi-Fi.

No resultado demonstrado na figura 4.7, ao utilizar a rede Wi-Fi o fluxo das mensagens é constante, verifica-se que não sofre grandes alterações, o que quer dizer que a aplicação consegue enviar os dados recolhidos pelos sensores sem atrasos. Isto significa que a rede Wi-Fi com a velocidade de upload de 20Mb pode também ser utilizada pela aplicação móvel do sistema MobileVJ.

4.2 Teste 2

Para o próximo teste foi utilizada a rede Evolved HSPA, uma vez que esta demonstrou bons resultados nos testes anteriores.

Durante o teste realizado desde o início estão a correr 2 dispositivos Android com todos os sensores ativados e um deles está a ler dados do dispositivo externo VJ. Os clientes de tipo subscriber serão adicionados com o tempo a decorrer.

O protocolo do teste Nº2 é demonstrado na Figura 4.8.

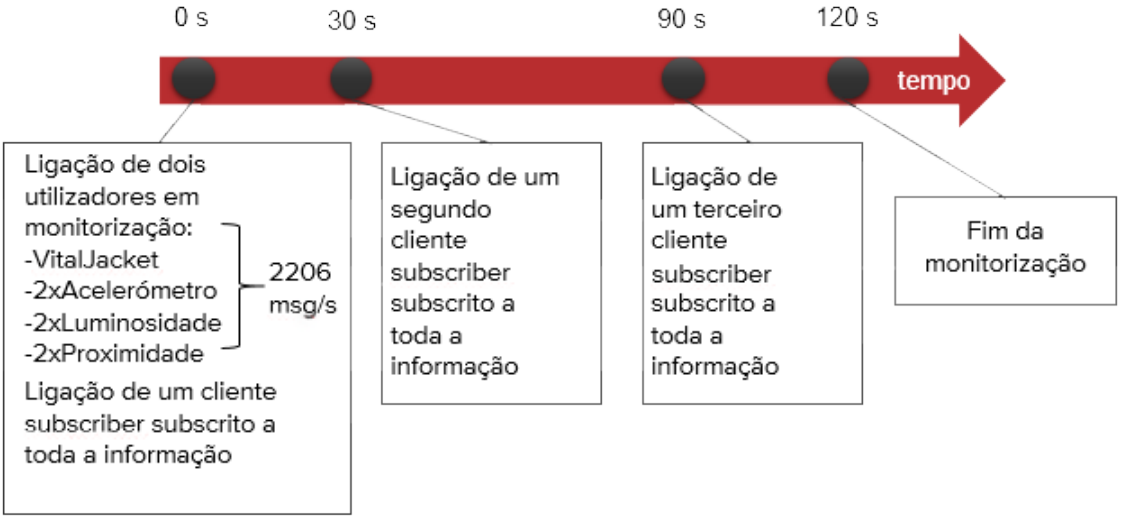

Figura 4.8: Protocolo do teste Nº2

Na Figura 4.9 a linha amarela representa o número de mensagens publicadas pelos dispositivos Android. A linha azul representa o número de mensagens enviado para as queues associadas aos clientes de tipo subscriber.

Como pode ser visto na Figura 4.9, o número de mensagens publicados é de 2206 mensagens/s. Na Figura 4.9 a) o número de mensagens publicadas e o número de mensagens entregues é equivalente. Passado um tempo a decorrer, foi adicionado mais um cliente, após um salto de fluxo de mensagens entregues este é estabilizado e após 5 segundos a média de mensagens entregues é

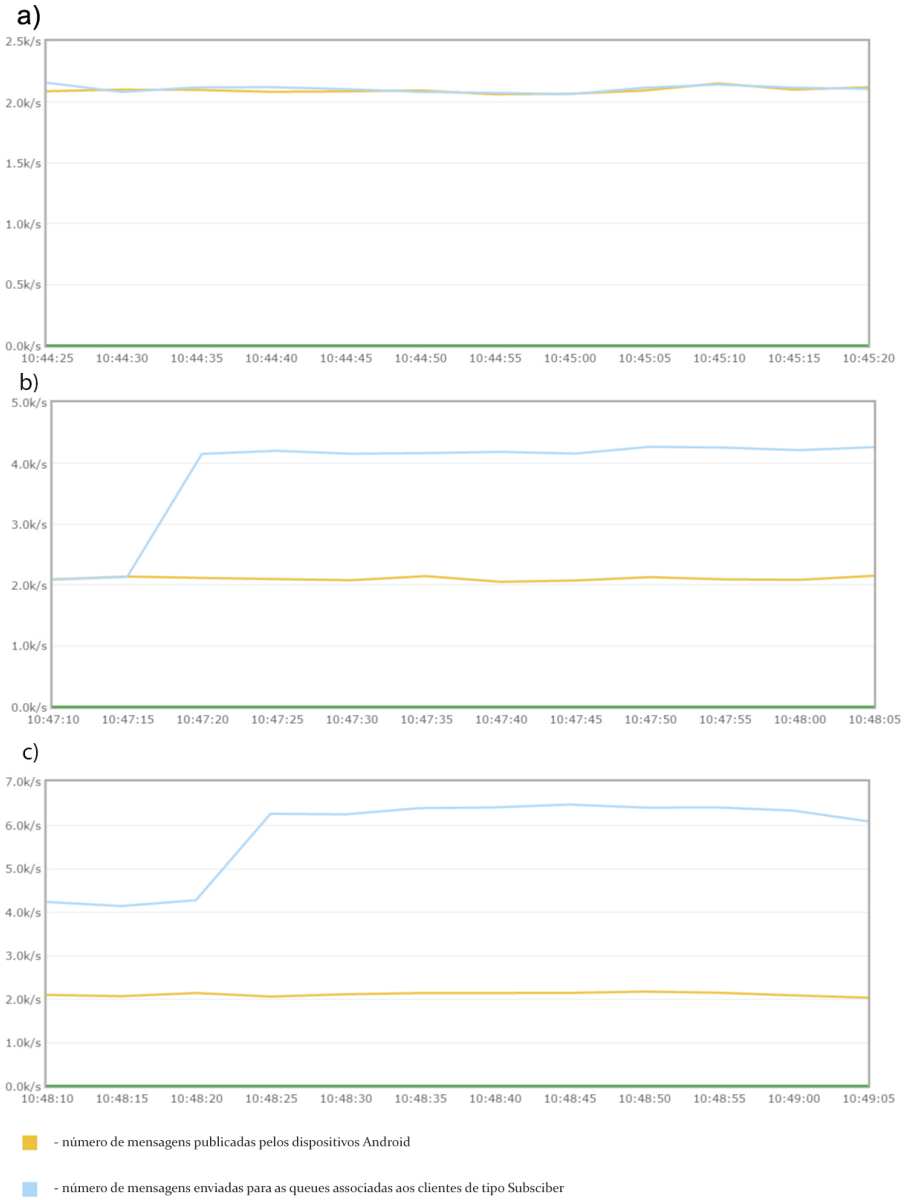

Figura 4.9: a) O sistema ligado com um cliente subscriber ativo. b) O sistema ligado com dois clientes subscriber ativos. c) O sistema ligado com três clientes subscriber ativos.

de 4400 mensagens/s (Figura 4.9 b)). Aquando do último cliente ser adicionado a média de mensagens entregues passa para 6604 mensagens/s(Figura 4.9 c)). Como se pode ver na Figura 4.9 o número de mensagens entregues é proporcional às mensagens publicadas. Verifica-se a regra que o número de mensagens entregues é igual ao Nº de mensagens publicadas x Nº de clientes.

Todos os clientes, durante a execução, comportaram-se de forma expectável. As aplicações publisher publicam em média 2206 mensagens/s. Foram recebidas 259600 em 259600 mensagens enviadas, isto quer dizer que não houve perdas de mensagens. Este é um bom resultado para um sistema de monitorização em tempo real. Destas 259600 mensagens o atraso médio foi de 1.5

segundo, que será um bom resultado para utilização de redes internet sem fios.

4.3 Realização de monitorização em condições reais

Para provar que através de dados recolhidos através do sistema MobileVJ consegue-se realizar a monitorização, foram realizados o teste Nº 3 e o teste Nº 4. Nos dois testes realizados foi utilizada a camisola VJ que recolheu os dados de ECG e Pulso.

Os dados recolhidos foram guardados num ficheiro *".txt"*, os resultados obtidos foram convertidos em dois gráficos por sensores correspondentes aos dois testes que podem ser analisados a seguir.

4.3.1 Test 3

No teste Nº3 foram recolhidos os dados de monitorização e este foi realizado com um indivíduo sentado durante 15 minutos. Figura 4.10.

Figura 4.10: Protocolo do teste Nº3

Durante a monitorização do teste Nº3, os gráficos (Figura 4.11) com os registos de dois sensores de atividade apresentam valores constantes devido à falta de movimento por parte do indivíduo monitorizado, o que acaba por refletir, tal como esperado, um batimento cardíaco relativamente baixo a oscilar entre os 62 e os 102 batimentos por minuto (bpm), apresentando uma média de 81.57 bpm

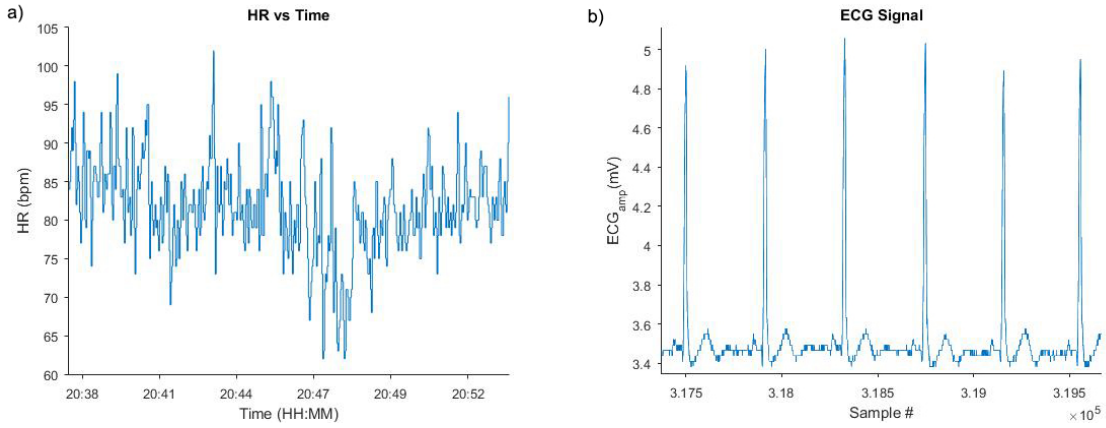

Figura 4.11: Valores recolhidos pelos sensores monitorizados no teste Nº3

4.3.2 Test 4

O teste Nº4 foi realizado durante a execução de uma série de um exercício físico (puxada de tríceps). Este durou 5 minutos, na qual foram realizadas 3 séries de 20 repetições(Figura 4.12).

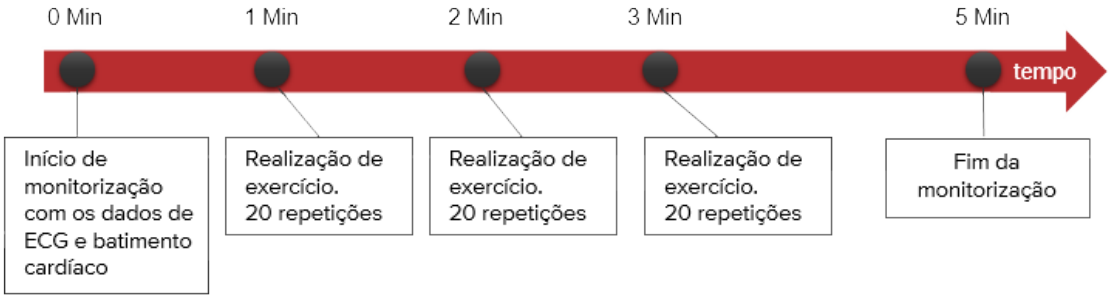

Figura 4.12: Protocolo do teste Nº4

No quarto teste, o sujeito examinado ao fazer três séries de exercício físico provoca alteração no comportamento do indivíduo. É possível verificar um aumento significativo na sua atividade cardíaca durante a realização do exercício e uma baixa da atividade cardíaca durante o tempo de descanso. O batimento cardíaco mínimo é de 67 bpm e este valor é obtido ainda antes de realizar o exercício físico. Quanto ao valor máximo é de 128 bpm. A média apresentada durante o período de monitorização é de 104.28 bpm.

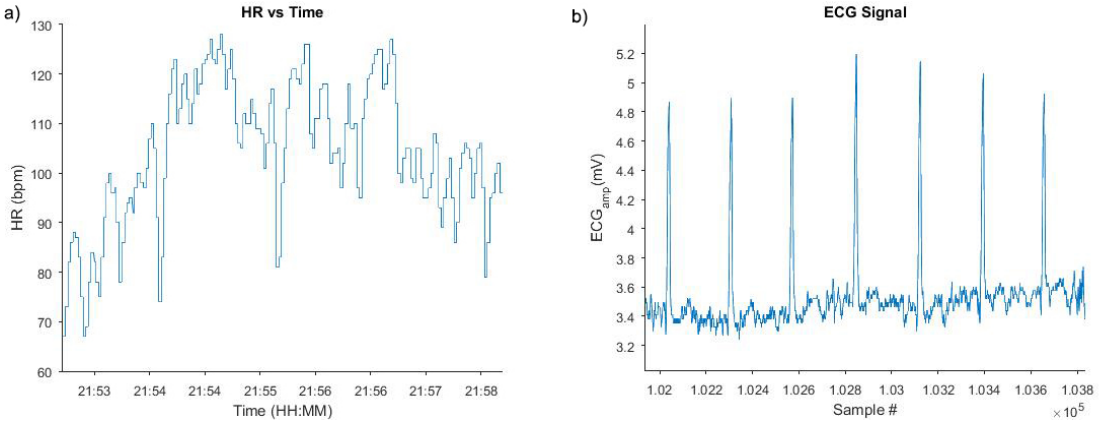

Figura 4.13: Valores recolhidos pelos sensores monitorizados no teste Nº4

4.4 Conclusão

Com o desenvolver do sistema MobileVJ foi necessário a realização de testes que comprovassem o funcionalidade de monitorização de vários utilizadores em tempo real. O ponto importante a testar seria saber em que circunstâncias a MobileVJ app não conseguiria enviar os dados recolhidos pelos sensores sem atrasos.

Como se pode constatar no primeiro teste, a rede EDGE é inapropriada para o uso na aplicação móvel do sistema MobileVJ, a velocidade média de upload que esta oferece é de 0.06Mb. Esta é insuficiente para que não haja atrasos na comunicação entre a aplicação móvel e o servidor. Isto quer dizer que a monitorização em tempo real seria impossível.

O segundo teste realizado com o Evolved HSPA demonstra que esta já é suficiente para o uso de aplicação móvel do sistema MobileVJ, pois não se encontram atrasos nas mensagens. A velocidade de upload média que esta oferece é de 3.42 Mbps.

O terceiro teste com a rede Wi-Fi utilizada demonstrou que esta oferecia velocidade média de upload de 20 Mbps. O fluxo das mensagens permanecia constante, e a sua velocidade de mensagens lidas por segundo era aproximadamente equivalente ao da rede Evolved HSPA com a velocidade de upload de 3.42 Mbps. Isto quer dizer que a velocidade suficiente para o uso de aplicação no dispositivo Android seria de 3.42 Mbps, que permite o envio de cerca de 4574 mensagens por segundo com um tamanho médio de 98 bytes, tendo em conta que só o VJ envia 2 mil mensagens por segundo.

O quarto e quinto testes realizados demonstram que o sistema MobileVJ é capaz de incluir mais de um cliente de tipo publisher, como também clientes de tipo subscriber. O número das mensagens entregues é igual a Nº mensagens publicadas x Nº de clientes de tipo Subsciber. O atraso médio entre a recepção de dados pelo cliente de tipo subscriber e recolha de dados pelo utilizador de tipo publisher é de 1.5 segundos. Este é um bom resultado para um sistema de

monitorização em tempo real para que se possa reagir de imediato caso um dos monitorizados necessite de cuidados de saúde imediatos. Os últimos dois testes demonstram que consegue-se obter os gráficos para uma análise através de dados recolhidos pelos indivíduos monitorizados que utilizam o sistema MobileVJ.

Com os últimos dois testes realizados conclui-se que o sistema MobileVJ pode já ser utilizado e que a aplicação MobileVJ app é capaz de agregar a informação de múltiplos sensores do dispositivo móvel, como também dos dispositivos externos. De forma a que não ocorram atrasos na monitorização verificou-se que as redes Evolved HSPA e Wi-Fi têm uma velocidade de mensagens por segundo recebidas pelo servidor bastante próximas e que o seu fluxo é similar. Com isto, foi verificado que o utilizador do dispositivo móvel precisará de utilizar a rede móvel com uma velocidade de upload média acima de 3Mbps, para que o envio dos seus dados não sofra atrasos, uma vez que isto seria de grande interesse para a monitorização na área de profissões de perigo como bombeiros ou militares.

Capítulo 5

Conclusões e Trabalho Futuro

Nesta secção são apresentadas as principais conclusões, como também a importância do estudo efetuado nesta dissertação. São, assim, apresentadas as conclusões com maior contribuição, como também as contribuições secundárias. Por fim, são ainda descritas algumas intenções de trabalho futuro.

5.1 Principais Conclusões

A pesquisa realizada ao longo desta dissertação exposta no Estado de Arte (Capítulo 2) demonstra a evolução e o desenvolvimento das tecnologias tais como o bluetooth e o smartphone. Com o aparecimento dos smartphones, tem evoluído o número de aplicações com o propósito de monitorização pois estes podem ser utilizados independente da sua localização.

Como este projeto visa ser utilizado para a monitorização de utilizadores de profissões de perigo, seria importante recolher ECG ou mesmo o batimento cardíaco, e como os dispositivos móveis de momento não incluem os sensores fiaveis de recolha destes dados, têm surgido no mercado diversos dispositivos vestíveis de monitorização com diferentes sensores incluídos, como é o caso do VitalJacket®, que quando associados a aplicações móveis tornam-se ferramentas bastante úteis para o acompanhamento do estado de saúde dos seus utilizadores.

Neste sentido e no âmbito do sistema de monitorização MobileVJ que pretende criar um sistema de monitorização em tempo real dos utilizadores, surgiu o tema desta dissertação que consiste no desenvolvimento de uma aplicação Android, cuja arquitetura foi desenhada para permitir a agregação de informação relativa aos sensores presentes num smartphone e no VJ, permitindo que o utilizador possa escolher os sensores que pretende utilizar, como também no desenvolvimento de uma aplicação cliente capaz de estar subscrito a vários tipos de dados e recebê-los em tempo real.

Como foi demonstrado na arquitetura desenhada, o sistema MobileVJ precisava de correr no mínimo em 2 dispositivos, um dos quais seria o dispositivo Android com a aplicação móvel desenvolvida, e o outro seria o computador onde estaria instalado o servidor rabbitMQ e a base de dados mongoDB, com a aplicação de cliente a correr. Desta forma, este sistema foi desenvolvido no âmbito desta dissertação.

Com os testes realizados ao longo deste estudo verificou-se que o sistema MobileVJ encontra-se pronto para poder ser inserido na área de monitorização de saúde. É de notar que o MobileVJ é um sistema de monitorização em tempo real, e um ponto importante destes sistemas é a conexão à internet. Foi, então, verificado que a conexão da internet da aplicação móvel desenvolvida deve possuir uma velocidade de upload por volta dos 3Mbps para que não ocorram atrasos no envio de mensagens, tendo em conta que só o VJ envia em média 2 mil amostras de ECG por segundo. Também é de notar que os sistemas M2M são adequadas para a monitorização de saúde em tempo real, pois a análise e distribuição dos dados pode ser realizada na parte do servidor de forma a diminuir os recursos utilizados pelos dispositivos móveis.

Também foi verificado que através deste sistema consegue-se obter dados fiéis enviados pelos sensores externos ligados pelo bluetooth ao dispositivo Android, para conseguir realizar a monitorização de indivíduos.

5.2 Trabalho Futuro

Apesar das aplicações publisher e subscriber terem se mostrado como ferramentas úteis no auxílio à monitorização remota e à funcionalidade do sistema desenvolvido, há várias melhorias possíveis a efetuar futuramente.

Uma das melhorias seria aperfeiçoar a interface da aplicação publisher, como também da aplicação subscriber.

O segundo melhoramento possível a realizar na aplicação de tipo publisher seria fazer a conexão com múltiplos dispositivos ligados por bluetooth à aplicação publisher, mas deve-se ter em atenção que os dispositivos que incluem a tecnologia Bluetooth LE não podem se conectar ao mesmo tempo que um dispositivo que não contenha esta tecnologia.

Apesar de existirem melhoramentos possíveis para o sistema MobileVJ, os objetivos propostos inicialmente para esta dissertação foram cumpridos. Desta forma foi criada uma aplicação móvel capaz de efetuar a agregação de informação recolhida por sensores incorporados no dispositívo Android e por dispositivos externos, e uma outra aplicação capaz de receber os dados que lhe interessam, recolhidos pela aplicação móvel.

Relativamente ao desenvolvimento futuro deste trabalho, prevêm-se boas perspetivas. Esta é uma dissertação de natureza inovadora, e espera-se que tenha um contributo considerável para a melhoria de sistemas atuais de monitorização remota, como também, exerça um forte impacto na área de monitorização de profissões de perigo.

Anexo A

A.1 Dados recebidos pelo cliente de tipo *subscriber*


```
1  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379162","
       val: ":"202.0"}}
2  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379162","
       val: ":"202.0"}}
3  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379163","
       val: ":"202.0"}}
4  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379164","
       val: ":"202.0"}}
5  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379164","
       val: ":"201.0"}}
6  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379165","
       val: ":"201.0"}}
7  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379165","
       val: ":"201.0"}}
8  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379165","
       val: ":"201.0"}}
9  {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379166","
       val: ":"201.0"}}
10 {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379166","
       val: ":"201.0"}}
11 {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379167","
       val: ":"201.0"}}
12 {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379167","
       val: ":"200.0"}}
13 {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379167","
       val: ":"201.0"}}
14 {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379167","
       val: ":"200.0"}}
15 {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379168","
       val: ":"200.0"}}
16 {"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379168","
       val: ":"200.0"}}
```

```
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379168","val: ":"200.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379169","val: ":"200.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379169","val: ":"200.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379169","val: ":"200.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379170","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379170","val: ":"199.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379170","val: ":"199.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379170","val: ":"199.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379171","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379171","val: ":"199.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379171","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379172","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379172","val: ":"199.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379172","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379173","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379173","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379174","val: ":"198.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379176","val: ":"197.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379177","val: ":"197.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379177","val: ":"197.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379178","val: ":"197.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379178","val: ":"197.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379179","val: ":"197.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379179","val: ":"196.0"}}
```

```
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379180","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379180","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379181","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379181","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379181","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379182","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379182","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379182","
    val: ":"196.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379183","
    val: ":"195.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379183","
    val: ":"195.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379183","
    val: ":"195.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379184","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379184","
    val: ":"195.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379185","
    val: ":"195.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379185","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379186","
    val: ":"195.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379187","
    val: ":"195.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379187","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379187","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379188","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379188","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379188","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379189","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379189","
    val: ":"193.0"}}
```

```
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379189","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379189","
    val: ":"194.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379190","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379190","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379190","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379191","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379191","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379191","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379192","
    val: ":"192.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379192","
    val: ":"192.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379192","
    val: ":"193.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379192","
    val: ":"192.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379193","
    val: ":"192.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379193","
    val: ":"192.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379193","
    val: ":"192.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379194","
    val: ":"191.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379194","
    val: ":"192.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379194","
    val: ":"191.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379195","
    val: ":"191.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379195","
    val: ":"191.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379196","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379196","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379197","
    val: ":"191.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379197","
    val: ":"190.0"}}
```

```
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379198","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379198","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379198","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379199","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379199","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379199","
    val: ":"190.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379200","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379200","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379200","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379201","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379201","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379201","
    val: ":"188.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379202","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379202","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379202","
    val: ":"189.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379203","
    val: ":"188.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379203","
    val: ":"188.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379203","
    val: ":"188.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379204","
    val: ":"188.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379204","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379204","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379205","
    val: ":"187.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379205","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379205","
    val: ":"186.0"}}
```

```
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379206","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379206","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379207","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379207","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379207","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379208","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379208","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379208","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379208","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379209","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379209","
    val: ":"186.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379209","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379210","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379210","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379210","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379211","
    val: ":"185.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379212","
    val: ":"184.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379213","
    val: ":"184.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379213","
    val: ":"184.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379214","
    val: ":"184.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379214","
    val: ":"184.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379215","
    val: ":"184.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379215","
    val: ":"184.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379216","
    val: ":"184.0"}}
```

```
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379216","val: ":"182.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379216","val: ":"182.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379217","val: ":"182.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379217","val: ":"181.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379217","val: ":"182.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379218","val: ":"181.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379218","val: ":"181.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379218","val: ":"181.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379219","val: ":"181.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379219","val: ":"181.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379219","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379219","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379220","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379220","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379220","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379221","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379221","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379221","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379222","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379222","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379222","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379222","val: ":"180.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379223","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379223","val: ":"179.0"}}
```

```
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379223","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379224","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379224","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379224","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379225","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379225","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379225","val: ":"178.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379226","val: ":"179.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379226","val: ":"178.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379227","val: ":"178.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379228","val: ":"178.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379228","val: ":"178.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379228","val: ":"178.0"}}
{"sensor":"ECG","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379229","val: ":"178.0"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379233","val: ":"156"}}
{"sensor":"1","androidId":"12ca9042ebc1bf61","values":{"valY":"-0.047","valZ":"9.826","time":"1467970379249","valX":"1.398"}}
{"sensor":"1","androidId":"12ca9042ebc1bf61","values":{"valY":"-0.057","valZ":"9.816","time":"1467970379251","valX":"1.407"}}
{"sensor":"1","androidId":"12ca9042ebc1bf61","values":{"valY":"-0.047","valZ":"9.807","time":"1467970379252","valX":"1.407"}}
{"sensor":"1","androidId":"12ca9042ebc1bf61","values":{"valY":"-0.057","valZ":"9.826","time":"1467970379253","valX":"1.417"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379254","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379255","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379255","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379256","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379257","val: ":"156"}}
```

```
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379257","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379258","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379258","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379261","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379262","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379264","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379265","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379265","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379266","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379267","val: ":"156"}}
{"sensor":"PULSE","androidId":"12ca9042ebc1bf61","values":{"time: ":"1467970379267","val: ":"156"}}
```

Bibliografia

[1] Inc. Headquarters Bluetooth SIG. Bluetooth. Bluetooth. Available: https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth, acedido a última vez em 22 de Janeiro de 2016.

[2] Inc. Headquarters Bluetooth SIG. Bluetooth. Bluetooth Devices. Available: https://www.bluetooth.com/what-is-bluetooth-technology/ bluetooth-devices, acedido a última vez em 22 de Janeiro de 2016.

[3] Inc. Headquarters Bluetooth SIG. Bluetooth Core Version 3.0 + HS specification. Available: https://www.bluetooth.com/specifications/ adopted-specifications, acedido a última vez em 22 de Janeiro de 2016.

[4] Inc. Headquarters Bluetooth SIG. Bluetooth. Low Energy. Available: https://www.bluetooth.com/what-is-bluetooth-technology/ bluetooth-technology-basics/low-energy, acedido a última vez em 23 de Janeiro de 2016.

[5] Joanne Gikas e Michael M Grant. Internet and Higher Education Mobile computing devices in higher education : Student perspectives on learning with cellphones, smartphones and social media. The Internet and Higher Education, 19:18– 26, 2013. Available: http://dx.doi.org/10.1016/j.iheduc.2013.06.002, doi:10.1016/j.iheduc.2013.06.002

[6] Maged N Kamel Boulos, Steve Wheeler, Carlos Tavares e Ray Jones. How smartphones are changing the face of mobile and participatory healthcare : an overview , with example from eCAALYX. BioMedical Engineering OnLine, 10(1):24, 2011. Available: http://www.biomedical-engineering-online. com/content/10/1/24, doi:10.1186/1475-925X-10-24.

[7] Number of mobile (cellular) subscriptions worldwide from 1993 to 2015. Available: http://www.statista.com/statistics/262950/ global-mobile-subscriptions-since-1993, acedido a última vez em 26 de Janeiro de 2016.

[8] Number of smartphone users* worldwide from 2014 to 2019. Available: http://www.statista.com/statistics/330695/ number-of-smartphone-users-worldwide/, acedido a última vez em 26 de Janeiro de 2016.

[9] Feida Lin. Operating System Battle in the Ecosystem of Smartphone Industry *. (2004):622–626, 2009. doi:10.1109/IEEC.2009.136

[10] Derick A Johnson e Mohan M Trivedi. Driving Style Recognition Using a Smartphone as a Sensor Platform. pages 1609–1615, 2011.

[11] Techopedia. Mobile Application. Available: https://www.techopedia.com/ definition/2953/mobile-application-mobile-app, acedido a última vez em 30 de Janeiro de 2016.

[12] Market share of smartphone OS of total smartphone installed base in 2013 and 2015. Available: http://www.statista.com/statistics/385022/ smartphone-worldwide-installed-base-operating-systems-share/ , acedido a última vez em 30 de Janeiro de 2016.

[13] Developers. Bluetooth Low Energy. Available: http://developer.android.com/ guide/topics/connectivity/bluetooth-le.html, acedido a última vez em 2 de Fevereiro de 2016.

[14] Developers. Sensors Overview. Available: http://goo.gl/E553cr, acedido a última vez em 3 de Fevereiro de 2016.

[15] Bluetooth Explained. Available: https://www.groupon.com/articles/ bluetooth-speaker-buying-guide, acedido a última vez em 22 de Janeiro de 2016.

[16] João Luís Gomes Abreu Fernandes. VitalX: Um módulo e uma aplicação móvel para um monitor vestível de sinais vitais. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2015.

[17] Zephyr Technology. Medtronic. Available: http://www.zephyranywhere.com/ products/hxm-bluetooth-heart-rate-monitor.

[18] Zephyr Technology. Zephyr HxM Smart User Guid.

[19] Jolla P Silva Cunha, Bernardo Cunha, William Xavier, Nuno Ferreira, Luis Meireles e S A Biodevices. Vital-Jacket R : A wearable wireless vital signs monitor for patients ' mobility in Cardiology and Sports. pages 1–2.

[20] Kickstarter. Campaign. About this project. Available: https://www.kickstarter.com/projects/458038473/ jambadoo-a-bluetooth-receiver-that-can-connect-to/ description.

[21] Utilizar um rato, teclado ou trackpad Bluetooth com o Mac. Available: https://support.apple.com/pt-pt/HT201171#howmany

[22] Jingjing Yang, Zhihui Wang e Xiao Zhang. An iBeacon-based Indoor Positioning Systems for Hospitals. 9(7):161–168, 2015.

[23] Paul Martin, Nicholas Grupen e Samuel Mu. Demo Abstract : An iBeacon Primer for Indoor Localization. pages 190–191, 2014.

[24] A. S. Tanenbaum and M. Van Steen, Distributed Systems: Principles and Paradigms, 2/E.2007.

[25] P. Software, “What RabbitMQ can do for you?,” 2015. [Online]. Available: https://www.rabbitmq.com/features.html. acedido a última vez em 23 de Janeiro de 2016.

[26] International Journal of Telemedicine and Applications Volume 2015 (2015), Article ID 373474, 11 pages http://dx.doi.org/10.1155/2015/373474

[27] M. N. Boulos, S. Wheeler, C. Tavares, and R. Jones, "How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from eCAALYX,"Biomed Eng Online, vol. 10, p. 24, 2011.

[28] C. Park, P. H. Chou, Y. Bai, R. Matthews and A. Hibbs, "An ultra-wearable, wireless, low power ECG monitoring system,"2006 IEEE Biomedical Circuits and Systems Conference, London, 2006, pp. 241-244

[29] "What Is Wearable Tech? Everything You Need to Know Explained."Wareable. N.p., n.d. Web. 22 July 2016.

[30] "Wearable Computing Background and Theory."Wearable Android Android Wear & Google Fit App Development (2015): 933. Web

[31] D. Shamah. (2014). Israeli ECG T-shirt monitors hearts, saves lives. Available: http://www.timesofisrael.com/israeli-ecg-t-shirt-monitors-hearts-saves-lives/, última

[32] "Apache Kafka E Big Data."Ciência E Dados. N.p., 08 Apr. 2016. Web. 21 July 2016.

[33] "O Que é O Apache Kafka – Amazon Web Services."Amazon Web Services, Inc. N.p., n.d. Web. 21 July 2016.

[34] "Nicholas Piël."» ZeroMQ an Introduction. N.p., n.d. Web. 22 July 2016.

[35] "ØMQ - The Guide.- By Pieter Hintjens, CEO of iMatix, Web. 22 July 2016.